In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.

While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.

UNIVERSITÉ DE MONTRÉAL

# ROUTAGE OPTIQUE DANS DES RÉSEAUX UTILISANT DES ROUTEURS LATINS

LAURENT MERIC

DÉPARTEMENT DE GÉNIE INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

MARS 2003

*Your file Votre référence*
*ISBN: 0-612-86419-7*
*Our file Notre référence*
*ISBN: 0-612-86419-7*

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

CE MÉMOIRE INTITULÉ:

# ROUTAGE OPTIQUE DANS DES RÉSEAUX UTILISANT DES ROUTEURS LATINS

présenté par: MERIC Laurent

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment acceptée par le jury d'examen constitué de:

M. CHAMBERLAND Steven, Ph.D., président

M. PIERRE Samuel, Ph.D., membre et directeur de recherche

M. PESANT Gilles, Ph.D., membre et codirecteur de recherche

M. GALINIER, Philippe, Ph.D., membre

À mes parents,

# REMERCIEMENTS

Je voudrais exprimer mes remerciements à mon directeur de recherche, le professeur Samuel Pierre, pour sa disponibilité, son support et son encadrement durant mon cursus en maîtrise.

L'expression de ma reconnaissance va également à mon codirecteur de recherche, le professeur Gilles Pesant, pour son dévouement, sa disponibilité, sa compétence, son support et ses commentaires tout au long de ce projet. Mes remerciements vont ensuite à tous les membres du LARIM qui ont toujours été très disponibles pour m'aider si nécessaire, Rolland, Désiré, Fabien et les autres.

Enfin, si ce mémoire a pu aboutir, c'est aussi grâce au soutien inconditionnel de mes parents et de mes amis à qui je désire exprimer ma gratitude pour leurs encouragements et leur support infaillible.

# RÉSUMÉ

L'essor actuel des télécommunications requiert des bandes passantes de plus en plus importantes. La transmission électrique, largement exploitée jusqu'à présent, ne permet plus de satisfaire les nouveaux besoins (imagerie médicale, vidéoconférences, ...). Les réseaux optiques, utilisant la lumière comme support de l'information, s'affirment comme étant l'avenir de la réseautique grâce aux débits permis par les fréquences optiques (jusqu'à plusieurs Térabits par seconde en théorie). Ces perspectives attrayantes posent cependant de nouveaux problèmes pour l'exploitation de ces nouveaux réseaux. En particulier, le routage et l'affectation de longueurs d'onde sont régis par de nouvelles contraintes. Nous nous proposons à travers ce mémoire de résoudre cette problématique dans le cas de réseaux comprenant des routeurs latins. Ces derniers sont bien plus robustes et beaucoup moins chers que des routeurs classiques. Ils introduisent cependant de nouvelles contraintes sur le routage et l'affectation de longueurs d'onde. Le problème ainsi défini est d'une de très grande complexité et ne peut être résolu dans un temps polynomial (en fonction du nombre de routeurs du réseau) et nous avons donc tenté de développer une heuristique pour réaliser un routage efficace dans de tels réseaux.

Nous avons choisi de développer un algorithme de recherche locale à voisinage variable (l'algorithme VNSFOR pour Variable Neighbourhood Search For Optical Routing). Les deux axes d'études ont donc été de définir des voisinages les plus performants possibles (au sens où ils ont la plus grande probabilité possible de contenir un bon voisin) et d'être capable de les explorer le plus rapidement possible. Pour la définition des voisinages, nous nous sommes basés sur l'interversion des connexions de ports (en entrée ou en sortie) des routeurs latins comme mouvement élémentaire. Chaque voisinage résulte de la combinaison d'un ou de plusieurs mouvements élémentaires.

L'algorithme ainsi créé parcourt donc des voisinages qui sont relativement basiques au départ (une seule interversion donc des voisins relativement proches) pour ensuite étudier des voisinages plus complexes qui ambitionnent de pouvoir s'extirper des maxima locaux (combinaison de plusieurs mouvements élémentaires). En ce qui concerne l'étude précise et rapide de ces voisinages, nous avons mis au point un re-routage incrémentiel en se basant sur l'effet des contraintes introduites par les routeurs latins pour minimiser le temps passé à estimer la qualité d'un voisin. Nous avons détaillé ainsi dans ce mémoire les résultats de notre algorithme par rapport à ceux d'un algorithme de référence de la littérature. Nous avons montré comment nous parvenons à apporter une amélioration de plus de 4% en moyenne, pour toutes les tailles de réseaux, au nombre de connexions obtenues pour un réseau aléatoire. De plus, notre algorithme présente la capacité de s'adapter de manière avantageuse à une demande précise de connexions : l'amélioration apportée est supérieure à 15% pour toutes les tailles de réseaux envisagées.

Nous nous sommes intéressés enfin à développer un générateur de réseaux plus réalistes. En effet, les réseaux tests générés aléatoirement doivent être le plus proche possible de la réalité, afin de valider pleinement nos algorithmes de routage. Nous avons donc créé des réseaux qui possèdent tous une ossature commune en anneau, en s'inspirant de ce qui se passe dans les réseaux réels. Nous avons montré alors comment ces réseaux apportent davantage de contraintes et constituent une plateforme de tests plus exigeante et plus sélective.

# ABSTRACT

The current rapid expansion of telecommunications requires increasingly large bandwidths. Electrical transmission, vastly exploited until now, does not permit to satisfy the new needs (medical imagery, videoconferences...). Optical networks, using light as data carrier, are now likely to be the future of networking thanks to the flow-rate permitted by optical frequencies (up to several TeraBits in theory). However, these attractive prospects bring new problems upon the exploitation of these new networks. In particular, the routing and wavelength assignment problem is governed by new constraints. We propose through this master's thesis to solve these problems in the case of networks including latin routers. The latter are much more robust and less expensive than traditional routers. However, they introduce new constraints on routing and wavelength assignment. As the problem defined is very complex and cannot be solved in a polynomial time (in function of the number of routers of the network), we tried to develop an heuristic to produce an effective routing in such networks.

We chose to develop an algorithm based on variable neighbourhood search (algorithm VNSFOR for Variable Neighbourhood Search For Optical Routing). The two directions of studies were to define the most efficient neighbourhood (where there is the greatest probability to find a good neighbour) and to be able to explore it quickly. For the definition of the neighbourhood, we based our studies on the inversion of ports connections (in entry or exit) of latin routers as elementary movement. Each neighbourhood results from the combination of one or more elementary movements. The algorithm created thus browses neighbourhoods which are relatively basic at the beginning (only one inversion on relatively close neighbours) and then studies more complex neighbourhoods which aim to escape from local maxima (combination of several elementary movements). With regard to the accurate and fast study of these

neighbourhoods, we developed an incremental re-routing based on the effect of the constraints introduced by latin routers, to minimize the time spent to estimate the quality of a neighbour. Then, we detail in this master's thesis the results of our algorithm compared to those of a reference algorithm of the literature, the LONCA algorithm [1]. We show how we manage, when following several scenarios, to provide better routings on the studied networks.

Finally, we focused on developing a generator of more realistic networks. Indeed, the randomly generated networks must be as close to feasible networks as possible, in order to validate accurately our routing algorithms. Thus we created networks which have a ring as common backbone, inspired by what occurs in real networks. Then we show how these networks bring more constraints and constitute a testing platform more demanding and more selective. We end finally by underlying the points which would deserve a deepening in the light of the results of this master's thesis.

# TABLE DES MATIÈRES

# LISTE DES FIGURES

# LISTE DES TABLEAUX

# LISTE DES ABRÉVIATIONS

| | |
|---|---|
| **ALLR** | Adaptative Least Load Routing |
| **DLE** | Dynamic Lightpath Establishment |
| **DSATUR** | Degree of Saturation |
| **FF** | First Fit |
| **FP** | Fixed Priority |
| **FPLC** | Fixed-Path Least-Congestion |
| **FR** | Fixed Routing |
| **LI** | Least Influence |
| **LL** | Least Loaded |
| **LLR** | Least Load Routing |
| **LONCA** | Local-search Optical Network Configuration Algorithm |
| **LR** | Latin Router |
| **MB** | Minimal Blocking |
| **MC** | Maximal sum of Chanel capacities |
| **MU** | Most Used |
| **MΣ** | Maxsum |
| **NP** | Nondeterministic Polynomial time |
| **NSFNET** | National Science Foundation Network |
| **PXC** | Photonic Cross-Connect |
| **RCL** | Relative Capacity Loss |
| **RD** | Random Distribution |
| **RLI** | Relative Least Influence |
| **RWA** | Routing and Wavelength Assignment |
| **SLE** | Static Lightpath Establishment |
| **VNSFOR** | Variable Neighbourhood Search For Optical Routing |
| **WPC** | Wavelength Path Capacity |
| **WRS** | Wavelength Routing Switch |

# CHAPITRE I
# INTRODUCTION

Les applications multimédia de la prochaine génération, comme les vidéoconférences ou l'imagerie médicale, requièrent un besoin sans cesse grandissant de larges bandes passantes. Les réseaux existants, utilisant la transmission électrique, imposent des limites de débit qui ne permettent plus de satisfaire les demandes actuelle et future. Il peut donc sembler judicieux de repenser complètement les réseaux de transmission à partir d'un nouveau moyen de communication : la transmission optique. En effet, les limites théoriques à plusieurs TeraBits par seconde nous laissent espérer que la conception de réseaux optiques puisse répondre largement à la demande. De plus, la transmission optique offre une grande fiabilité, de faibles pertes (jusqu'à seulement 0.2 dB/km), une immunité aux perturbations électromagnétiques et une grande transparence vis à vis des protocoles implémentés. L'objectif de ce mémoire est l'étude des contraintes particulières imposées par des routeurs spécifiques, les routeurs latins, dans les réseaux optiques. Attractifs non seulement par leur prix mais aussi par leur grande robustesse, ils introduisent cependant de fortes relations entre le chemin choisi pour relier deux nœuds au travers du réseau et la longueur d'onde qui lui est associée. La détermination du routage dans de tels réseaux est la difficulté majeure de leur conception et de leur utilisation. La finalité de ce travail est donc de mettre en place un algorithme qui résolve efficacement les problèmes de routage dans des réseaux optiques utilisant des routeurs latins malgré la grande complexité des problèmes rencontrés. Dans ce chapitre d'introduction, nous présentons d'abord quelques concepts de base et des éléments de la problématique, avant de résumer nos objectifs de recherche, les résultats escomptés et le plan du mémoire.

## 1.1 Définitions et concepts de base

Comme toute nouvelle technologie, la transmission optique possède un ensemble de définitions et de concepts qui lui sont propres. Nous nous devons de les introduire ici, notamment pour faire la distinction avec toutes les notions précédemment mises en place pour la transmission électrique. En effet, un *réseau optique* est un ensemble de nœuds qui utilisent la lumière comme support de l'information pour communiquer entre eux. L'information est véhiculée entre ces nœuds par des fibres optiques qui se comportent comme des guides d'ondes lumineuses et qui permettent l'acheminement de données optiques depuis l'origine jusqu'à la destination. Le transfert s'effectue sur une longueur d'onde donnée parmi les longueurs d'onde permises par les entités communicantes. On appelle *routage optique* la sélection de la suite de nœuds du réseau à travers lesquels l'information va être acheminée pour rejoindre la destination finale; les nœuds intermédiaires sont alors dénommés *routeurs optiques*. Cette suite de nœuds du réseau constitue un *chemin optique*. La spécificité majeure de la transmission optique est le multiplexage fréquentiel qui y est réalisé : les longueurs d'onde accessibles sur une fibre constituent des « canaux » de transmission indépendants sur lesquels le nœud émetteur peut transmettre simultanément des flots de données indépendants.

L'évaluation de la performance d'un routage optique peut être effectuée suivant différentes approches. Elle peut par exemple se focaliser sur le nombre maximal de connexions qu'il établit. Elle peut aussi estimer sa souplesse face à un événement extérieur (panne, demande d'une connexion supplémentaire). Le terme de *survivabilité* (cas d'une panne) se réfère alors à la capacité d'un réseau à offrir les mêmes possibilités de transmission aux nœuds du réseau malgré la rupture d'une fibre optique ou même d'un routeur. Une *panne simple* correspond au bris d'une seule entité du réseau optique (lien ou nœud), alors que des *pannes multiples* se réfèrent à des bris multiples. *L'efficacité du routage* (cas d'une connexion supplémentaire) dans un réseau optique est mesurée par la probabilité de blocage du réseau. Comme le laisse présager son nom, elle évalue la probabilité qu'une tentative d'établissement d'un nouveau chemin optique échoue à cause du routage optique précédemment choisi.

Nous présentons ici les différentes composantes de la conception de réseau optique pour situer notre travail parmi l'ensemble des recherches menant à la résolution du problème global.

La première étape consiste en une détermination du trafic que le réseau va devoir supporter. Aussi étrange que cela puisse paraître, cette étape comporte une complexité très importante. Elle se doit d'évaluer l'origine, la destination, le type, l'application, l'agrégation, les disciplines de priorité et les types de contrôle du trafic. De plus, le réseau établi se devant de présenter une certaine durée de vie, cette étape requiert une estimation du trafic futur.

Une fois le trafic connu, nous nous devons de choisir une plate-forme qui supportera le trafic. Il convient alors d'analyser les restrictions imposées par la plate-forme au niveau architecture, performance et protocole.

L'étape suivante constitue véritablement la partie de conception de réseau. Elle comprend une phase de design topologique et une phase de routage et dimensionnement. De par la complexité de la conception de réseau dans son ensemble, le problème est souvent divisé en deux : configuration topologique et synthèse de réseau. Il faut cependant reconnaître que toute division a priori d'un problème d'optimisation implique une perte potentielle d'optimalité.

La configuration topologique s'attache à proposer une position pour les nœuds et les liens du réseau, ainsi que les caractéristiques de ces équipements (performance des routeurs, capacité des liens, ...). Elle réalise une minimalisation du coût des liens et des équipements, sujet à des contraintes de topologie dorsale voulue, de sites potentiels des équipements, de satisfaction de demande de trafic et de restriction de routage.

Le problème de routage et dimensionnement, appelé aussi problème de « synthèse de réseaux », détermine les chemins optiques empruntés par les paquets optiques entre les paires origine/destination et leur affecte une longueur d'onde. Ces affectations doivent se faire de manière à répondre à une demande précise de trafic ou à minimiser la probabilité de blocage d'une transmission future.

La planification de réseau s'achève par une évaluation de la performance et des coûts du réseau. Celle-ci se réalise au moyen de modèles analytiques et de simulations qui testent les choix effectués.

Notre étude se focalise plus précisément sur la partie routage et affectation de longueurs d'onde et constitue donc un élément essentiel de la planification de réseau.

## 1.2 Éléments de la problématique

Nous exposons dans cette section tous les problèmes rencontrés lors de la résolution du routage dans un réseau optique, en mettant en lumière leur complexité. Cela permet de justifier l'intérêt porté par la science, et le nôtre, sur ce sujet.

### 1.2.1 Routage et affectation de longueurs d'onde

La contrainte majeure induite par le choix de la transmission optique est l'utilisation de longueurs d'onde différentes pour deux communications distinctes utilisant le même lien. De plus, comme nous le développerons plus avant dans ce mémoire, nous nous sommes intéressés à des réseaux n'incluant pas de convertisseurs de longueurs d'onde (à cause de leur prix et des restrictions qu'ils imposent en terme d'efficacité). Ce choix implique une seconde contrainte très importante : la contrainte de continuité de longueur d'onde. Un chemin optique occupe la même longueur d'onde sur chaque lien de son parcours. En considérant les deux contraintes simultanément, nous pouvons établir que deux chemins optiques possédant au moins un lien en commun doivent être transmis à des longueurs d'onde distinctes.

Le problème des choix des chemins optiques à établir et d'affectation des longueurs d'onde est très complexe, chacun étant NP-difficile pris séparément. Nous allons illustrer ceci dans le réseau optique à quatre nœuds de la Figure 1.1. Les nœuds représentent les quatre routeurs optiques du réseau, alors que les arêtes en gras correspondent à des liens physiques bidirectionnels entre les routeurs concernés.

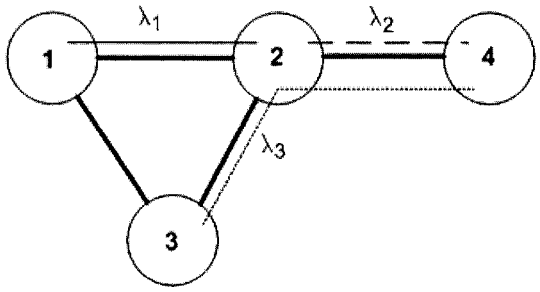
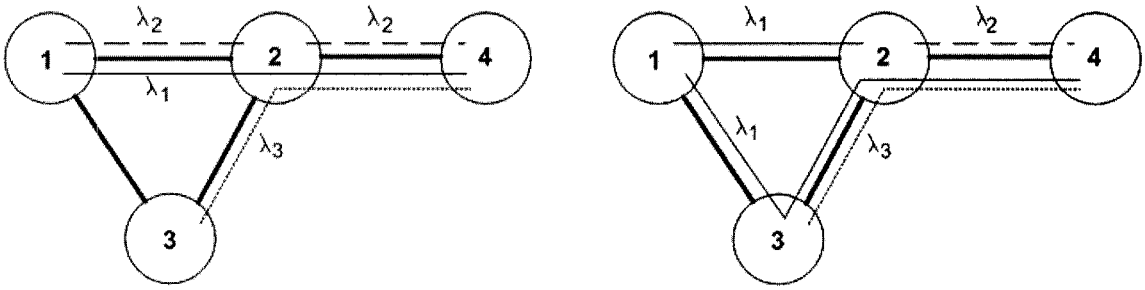



**Figure 1.1** Exemple de routage optique

Nous supposons que chaque lien possède trois longueurs d'onde disponibles (représentées sur la figure par les traits pleins, les tirets et les pointillés). Nous considérons aussi trois chemins précédemment routés comme sur le schéma (chemins 1-2, 2-4 et 3-2-4). Supposons que nous désirons satisfaire une requête entre les nœuds 1 et 4, en admettant que le routage nous impose de passer seulement par le nœud 2 comme nœud intermédiaire. Nous remarquons que la demande ne peut être satisfaite sans violer une des deux contraintes précédemment établies.

À la Figure 1.2, nous montrons que l'établissement du chemin 1-4 est possible : soit en changeant de stratégie d'affectation de longueur d'onde (on affecte la deuxième longueur d'onde à la connexion 1-2, schéma de gauche), soit en changeant de stratégie de routage (on se permet de passer par le routeur 3, schéma de droite).




Nouvelle stratégie d'affectation de longueurs d'onde

Nouvelle stratégie de routage

**Figure 1.2** Nouvelles stratégies satisfaisant la requête

Le problème global représente donc réellement un problème conjoint de routage et d'affectation de longueurs d'onde. La majorité des approches sur le sujet consistent à choisir une stratégie basique pour l'un des deux problèmes afin d'optimiser l'autre tout en cherchant à se rapprocher le plus possible de l'optimum global.

### 1.2.2 Les routeurs latins

Nous avons décidé d'étudier des réseaux qui utilisent des routeurs spécifiques, les routeurs latins, pour établir des liaisons entre différentes paires origine/destination. Ce choix est motivé par le prix très attractif de ces éléments, et par leur grande robustesse. En effet, ces routeurs intègrent des Waveguide Grating Routers (ou WGR) [2] comme sous-composants optiques de routage qui effectuent un routage totalement passif et sont donc plus fiables que des routeurs actifs. Cependant, les routeurs latins imposent énormément de relations liant le routage à l'affectation de longueurs d'onde et nous obligent à résoudre conjointement les deux problèmes. Nous donnons ici un aperçu des contraintes spécifiques des routeurs latins mais elles seront exposées en détail dans le chapitre suivant. Elles sont représentées sous la forme d'une matrice qui fait correspondre à chaque port d'entrée et de sortie des routeurs une longueur d'onde (assimilée ici à une lettre) à laquelle doit être effectuée la transmission. Cette matrice est une matrice de carré latin : une longueur d'onde donnée apparaît une fois et une seule dans chaque ligne et chaque colonne. Un exemple de matrice de routeur latin est représenté à la Figure 1.3.

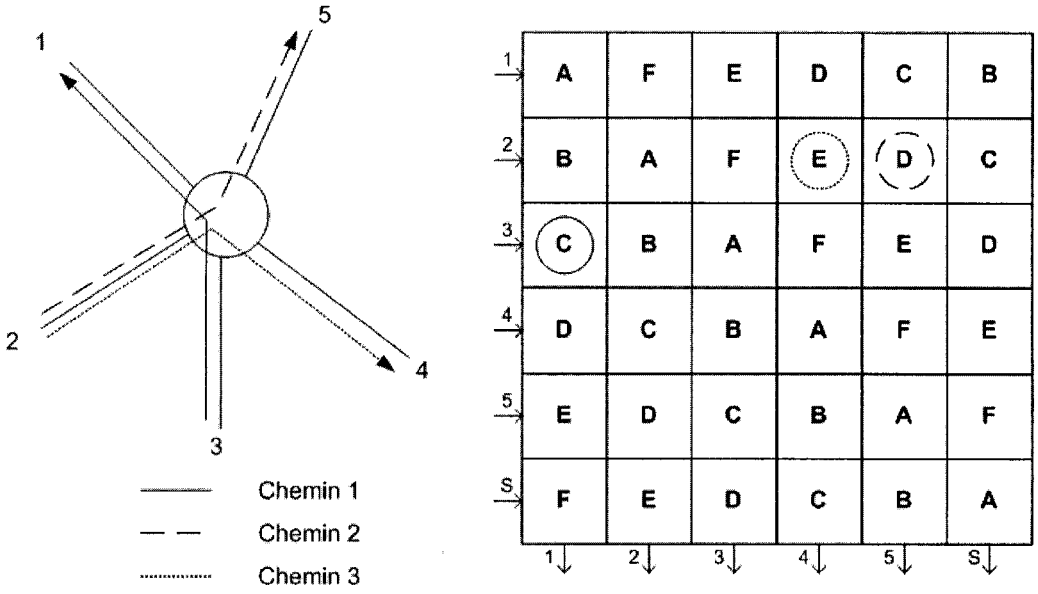



**Figure 1.3** Exemple de routeur latin

Les ports physiques sont représentés par des nombres (sauf le dernier qui est noté par la lettre $S$ : c'est un port qui n'est jamais connecté et qui correspond toujours à un port terminal, initial ou final, d'un chemin optique). D'après les propriétés des routeurs latins, le chemin entrant par le port 3 et sortant par le port 1 (correspondant au chemin 1 ici) doit forcément être supporté par la longueur d'onde $C$.

Nous vous présentons à la Figure 1.4 un exemple de routage mettant en jeu des routeurs latins. Nous considérons trois routeurs latins de taille 3*3. La connexion des ports a déjà été réalisée. Les liens schématisés ici sont unidirectionnels (un port de sortie, s'il est connecté, est automatiquement relié à un port d'entrée). Cependant, s'il y a une liaison reliant un port de sortie du routeur $i$ à un port d'entrée du routeur $j$, il y a automatiquement une liaison entre un port de sortie de $j$ et un port d'entrée de $i$. En effet, les liaisons reliant véritablement deux routeurs sont bidirectionnelles.

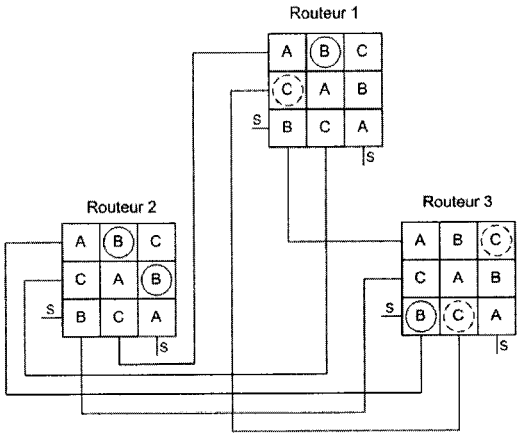



**Figure 1.4** Réseau de routeurs latins

Nous avons représenté les chemins optiques émis pas le routeur 3. Le routeur 3 possède un chemin optique avec le routeur 2 (sur la longueur d'onde *B*) et un chemin le reliant à lui-même (longueur d'onde *C*). Nous désirons réaliser la connexion depuis le nœud 3 jusqu'au nœud 1. Dans cette configuration, la requête ne peut pas être satisfaite. Cependant, nous remarquons que si nous invertissons les connexions des deux ports de sortie du routeur 3, nous sommes alors en mesure de connecter directement 3 avec 1, comme le montre la Figure 1.5.

Pour un cas extrêmement basique comme celui-ci, nous voyons déjà que les branchements à réaliser ne sont pas aisés. Nous désirons étudier ce routage pour des réseaux beaucoup plus grands, avec des routeurs de taille plus importante.

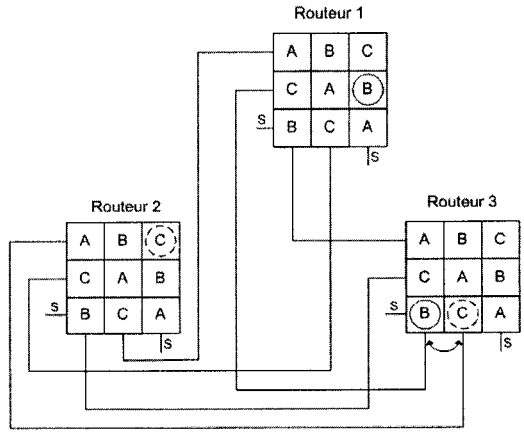



**Figure 1.5** Configuration efficace

## 1.3 Objectifs de recherche

Nous venons de mettre en exergue les difficultés que soulevaient le problème de routage dans un réseau optique composé de routeurs latins. Nous présentons ici les objectifs que nous nous sommes fixés à la lumière de ces défis.

Nous avons choisi de développer une approche basée sur la recherche locale pour résoudre le problème. Initialement, nous avons tenté de combiner recherche locale et programmation par contraintes (contrainte de continuité de longueur d'onde, contrainte de longueurs d'onde distinctes pour deux chemins optiques empruntant un lien en commun et contraintes imposées par l'utilisation de routeurs semblant jouer un rôle essentiel dans le routage optique) mais les résultats se sont avérés relativement infructueux. Nous avons alors porté notre attention sur l'optimisation des voisinages de la recherche locale pour proposer une solution au problème proposé.

Ce mémoire se donne donc pour objectif principal la mise au point d'une méthode heuristique de type recherche locale à voisinage variable pour résoudre le problème de routage et d'affectation de longueurs d'onde dans un réseau optique composé de routeurs latins. Cette méthode sera alors testée sur de nombreuses configurations de réseaux afin d'estimer la généralité de l'algorithme et afin de le comparer à des algorithmes connus trouvés dans la littérature.

## 1.4 Plan du mémoire

Après ce chapitre d'introduction, nous proposons dans le deuxième chapitre un recensement des travaux précédemment réalisés dans le cadre d'études sur le routage de réseaux optiques. Nous analysons plus particulièrement les recherches traitant les réseaux comprenant des routeurs latins, afin de dégager clairement la problématique de notre sujet. Dans le troisième chapitre, nous présentons la mise en œuvre de notre algorithme VNSFOR (pour Variable Neighbourhood Search For Optical Routing) qui tente de répondre efficacement à la problématique précédente. Dans le quatrième chapitre, nous exposons le protocole expérimental que nous avons suivi pour tester notre algorithme ainsi que les résultats auxquels nous avons abouti. Dans le cinquième

chapitre, nous abordons succinctement quelques réflexions qui ont guidé notre recherche. Nous achevons alors ce mémoire par une conclusion rappelant notre contribution au problème de routage optique en présence de routeurs latins et les extensions possibles à notre travail.

# CHAPITRE II
# ROUTAGE ET AFFECTATION DE LONGUEURS D'ONDE DANS UN RÉSEAU OPTIQUE

Le problème de routage et d'affectation de longueurs d'onde consiste à déterminer un patron d'affectation de chemins optiques et de longueurs d'onde aux paires origine/destination d'un réseau optique dans le but de maximiser le nombre de connexions réalisables (ou minimiser la probabilité de blocage), tout en respectant un certain nombre de contraintes imposées par les routeurs et les fibres optiques. Dans ce chapitre, nous faisons une revue sélective des travaux recensés dans la littérature qui traitent d'un ou de plusieurs aspects de ce problème. Après avoir introduit le cadre dans lequel se placera notre étude, nous présenterons des solutions déjà connues pour le problème global puis nous donnerons une description plus précise des travaux réalisés pour des réseaux utilisant des routeurs latins. Nous terminerons enfin en introduisant la recherche locale à voisinage variable.

## 2.1 Contexte et concepts de base de notre étude

L'existence de routeurs optiques a rendu possible la commutation optique. Ainsi, cette étude ne traite que des réseaux totalement optiques qui n'ont pas recours en leur sein à la conversion électro-optique. Cela se justifie aisément par la volonté d'exploiter réellement la bande passante allouée par la transmission optique en s'affranchissant ainsi des limites imposées par la transmission électrique. De plus, les convertisseurs de longueurs d'onde ne sont pas utilisés. En effet, ils sont extrêmement onéreux et ont recours à la conversion électro-optique, ce qui peut présenter une perte d'efficacité au niveau de la transmission. Ainsi, la restriction majeure de cette étude est la contrainte de continuité de longueur d'onde : un chemin optique (un ensemble de liens dans le réseau reliant l'origine et la destination d'une transmission optique) doit occuper la même

longueur d'onde sur chaque lien de son chemin. Sachant que deux chemins lumineux distincts ne peuvent emprunter le même lien avec la même longueur d'onde, l'objectif du problème est le routage et l'affectation de longueurs d'onde aux chemins du réseau. Ceci doit répondre à une requête donnée (matrice de trafic ou exploitation maximale du réseau) et doit satisfaire les contraintes imposées par le routage optique (l'utilisation unique d'une longueur d'onde sur un lien et la continuité de longueur d'onde le long d'un chemin optique, tout en respectant la capacité de tous les liens, i.e. le nombre de longueurs d'onde allouées sur chacun de ses liens). Ce problème est nommé RWA dans la littérature (pour Routing and Wavelength Assignment).

Il est cependant à noter que l'on trouve aussi dans la littérature des recherches qui se permettent d'utiliser quelques convertisseurs de longueurs d'onde dans le réseau. Elles tentent alors de minimiser leur nombre (donc leur coût). L'approche adoptée est cependant complètement différente de celle développée dans ce mémoire, étant donné qu'elle modifie les contraintes prises en compte. Notre approche est toutefois légitimée par les résultats actuels qui montrent que l'efficacité apportée par les convertisseurs est moindre comparée aux coûts qu'ils entraînent [18].

Avant d'aller plus avant dans la présentation du problème de routage et d'affectation de longueurs d'onde dans un réseau optique, il faut d'abord souligner qu'il y a deux approches différentes : statique ou dynamique [26].

- Dans le cas statique, la matrice de trafic est connue une fois pour toute. On parle d'établissement de chemin optique statique (SLE pour Static Lightpath Establishment).
- Dans le cas dynamique, on considère l'arrivée de requêtes de connexion et la disparition de chemins déjà établis. Il s'agit d'établissement de chemin optique dynamique (DLE pour Dynamic Lightpath Establishment).

Le cas dynamique est beaucoup plus complexe puisqu'il requiert un maintien d'informations concernant l'état du réseau. Cependant, nous nous focaliserons dans notre étude plus particulièrement sur le cas statique.

Finalement, la difficulté consiste à optimiser globalement les problèmes de routage et d'affectation de longueurs d'onde, chacun étant NP-difficile pris séparément. Nous présentons donc les principaux algorithmes rencontrés dans la littérature qui tentent de répondre à cette question.

## 2.2 Méthodes classiques de résolution

Etant donné la complexité des problèmes mis en jeu, les algorithmes exacts souvent basés sur une énumération de toutes les possibilités ne peuvent être appliqués qu'à des problèmes de petite taille. Pour des problèmes de taille plus conséquente, il devient nécessaire de mettre en œuvre des heuristiques pour obtenir des résultats satisfaisants dans des temps raisonnables. Devant ce double problème RWA, nous pouvons distinguer deux types d'approches. En effet, certains auteurs apportent une attention toute particulière aux algorithmes de routage en conservant une stratégie basique pour l'affectation de longueurs d'onde, alors que d'autres s'attachent, à l'inverse, à optimiser leurs algorithmes d'affectation de longueurs d'onde quitte à utiliser des stratégies moins optimisées pour le routage comme des algorithmes connus de plus courts chemins.

### 2.2.1 Algorithmes connus de routage

Les stratégies classiques de routage statique sont basées sur des algorithmes de plus court chemin comme les algorithmes de Dijkstra ou Bellman-Ford. Elles ont pour but de dresser une table de routage pour chaque nœud du réseau en occultant la stratégie d'affectation de longueurs d'onde qui sera appliquée par la suite. Une revue de différentes stratégies de routage est réalisée en [6], où les auteurs comparent les performances respectives de ces algorithmes sur le réseau optique européen. Ils examinent plus particulièrement les algorithmes de routage du plus court chemin et de routage du chemin à plus petit nombre de sauts.

**Routage du plus court chemin**

Cette technique affecte à une paire de nœuds du réseau le chemin le plus court en terme de distance. Elle est réalisée au moyen d'un algorithme de plus court chemin (par exemple Dijkstra) appliqué au graphe composé des nœuds du réseau, où les liens représentent les connexions entre deux routeurs. Le poids d'un lien est alors fixé à la distance séparant les deux nœuds qu'il relie. L'intérêt de cette stratégie est de minimiser la longueur des connexions réalisées et de minimiser ainsi, après affectation des longueurs d'onde, la longueur totale de fibres utilisées. Ceci pourra se révéler intéressant pour l'étude de réseaux étendus avec de grandes distances entre les nœuds.

**Routage du chemin à plus petit nombre de sauts**

La stratégie LNH [6] (ou Least Number of Hops) minimise le nombre de sauts pour les paires de nœuds communicantes du réseau, où un saut correspond à un routeur traversé au cours de la transmission optique. Cette méthode est effectuée à nouveau au moyen d'un algorithme de plus court chemin où le poids des liens est fixé à 1. Ainsi, la longueur d'un chemin correspond au nombre de routeurs traversés jusqu'à la destination, en exceptant le routeur origine (nombre de sauts du chemin moins un). Cette stratégie convient en particulier pour les cas où l'on désire minimiser les délais de transmission engendrés par les routeurs intermédiaires. De plus, en minimisant le nombre de liens pour les chemins optiques, elle peut permettre de préserver les ressources de longueurs d'onde dans le réseau et faciliter ainsi l'attribution de longueurs d'onde.

**Résultats sur le réseau européen**

Pour tester les algorithmes de routage, les auteurs allouent aux chemins déterminés la première longueur d'onde accessible parmi une liste prédéfinie. Les tests sont effectués sur le réseau optique européen composé de 17 nœuds et 72 liens unidirectionnels. Pour juger de la performance de ces algorithmes de routage, deux études sont réalisées :

- calculer le nombre de longueurs d'onde nécessaires par lien (pour que chaque paire de nœuds soit reliée par un chemin optique (une seule fibre par lien, cas 1));
- calculer le nombre de fibres (composées de 8 longueurs d'onde chacunes) à installer dans le réseau pour que chaque paire de nœuds soient reliée par un chemin optique (plusieurs fibres par lien, cas 2).

Les résultats fournis par les auteurs sont compilés dans le Tableau 2.1.

**Tableau 2.1** Comparaison des algorithmes de routage

| | | Plus court chemin | Chemin à plus petit nombre de sauts |
|---|---|---|---|
| cas 1 | Nombre de longueurs d'onde par lien | 31 | 27 |
| cas 2 | Nombre de fibres | 167 | 158 |
| | Longueur totale de fibres (km) | 97430 | 109275 |

L'algorithme de chemin à plus petit nombre de sauts consomme moins de ressources en terme de longueurs d'onde. Cependant, il nécessite une plus grande longueur de fibres optiques. De nombreuses recherches sur le routage optique statique s'inspirent de ces algorithmes de base en conservant les calculs de plus courts chemins mais en optimisant les poids affectés à chaque lien. De plus, des améliorations sont généralement apportées pour affecter au fur et à mesure les longueurs d'ondes aux chemins déjà crées.

**Plus courts chemins alternatifs**

Une étude sur les performances de méthodes de plus courts chemins pour résoudre le problème de routage optique est proposée en [21]. Elle analyse des réseaux à nombre fixé de longueurs d'onde par fibre, avec une seule fibre par lien optique. À cause de la contrainte de continuité de longueur d'onde, les chemins optiques, basés

exclusivement sur une méthode de plus court chemin, ne peuvent pas être établis simultanément entre chaque paire de nœuds. Cette limitation a poussé l'auteur à explorer l'idée de chemin alternatif (qui est court sans être le plus court possible) dans le cas où le plus court chemin est bloqué par la contrainte de continuité de longueur d'onde. Les différents chemins alternatifs sont définis comme suit :

- le plus court chemin est le chemin SP-1;
- la meilleure alternative possible à SP-1 est le second chemin le plus court SP-2;
- le chemin SP-3 est le plus court chemin suivant SP-2, et ainsi de suite;
- le chemin SP-N est le $N^{ème}$ plus court chemin (si l'on considère que les chemins sont classés par ordre de longueur croissante).

Il est à noter qu'il n'y a aucune contrainte d'indépendance entre les différents chemins alternatifs (pas de restrictions sur le nombre de liens ou de routeurs en communs). Les chemins successifs sont donc plus difficiles à trouver car il ne suffit pas d'ôter les liens du plus court chemin du graphe pour trouver le second chemin le plus court.

La méthode proposée pour trouver le deuxième plus court chemin reliant $s$ à $d$ dans un graphe donné est exposée à la Figure 2.1 .Pour trouver le chemin SP-2, elle génère tout d'abord le chemin SP-1 (algorithme de Dijkstra appliqué à $s$ jusqu'à ce que $d$ entre dans la liste des sommets dont la distance à $s$ est fixée). Elle ôte ensuite un lien de ce chemin (coût infini dans le graphe) et relance la procédure de recherche d'un chemin SP-1 dans ce nouveau graphe. Le chemin obtenu est stocké dans une liste mémoire TP. En répétant la procédure sur tous les liens du chemin SP-1 original, le chemin SP-2 est le plus court de la liste TP. Les chemins alternatifs successifs sont obtenus par récurrence à partir du chemin précédent. Ainsi, pour créer le chemin SP-3, il faut remplacer SP-2 par SP-3 et SP-1 par SP-2 dans l'exemple proposé. Il est à préciser que si deux chemins ont même longueur, l'un des deux est choisi aléatoirement comme plus court chemin.

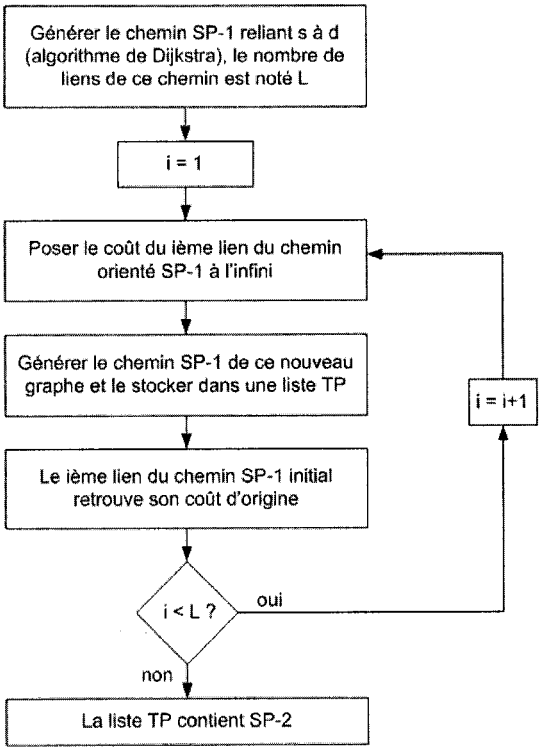



**Figure 2.1** Algorithme pour exhiber le SP-2

L'auteur [21] a décidé de comparer les algorithmes du plus court chemin pour N (nombre de plus courts chemins alternatifs cherchés) allant jusqu'à 3. L'optimisation cherche à satisfaire du mieux possible une matrice de trafic donnée (indiquant la taille de l'information à faire transiter entre chaque paire de nœuds). Les algorithmes doivent donc minimiser l'information non assignée par le routage réalisé. Pour tenir compte d'une stratégie d'affectation de longueurs d'onde, l'auteur définit, par rapport aux algorithmes de plus court chemins précédents, les trois algorithmes de routage suivant :

- l'algorithme RWA-1 ordonne les demandes de la matrice de trafic en ordre décroissant. Il prend la demande maximale non encore traitée et cherche le chemin SP-1 correspondant. Si aucune longueur d'onde n'est disponible tout au long du chemin, le chemin est bloqué, l'information correspondante est considérée comme non assignée et la demande est marquée comme traitée. Si l'ensemble des longueurs d'onde disponibles est non vide, la plus petite d'entre

elles est assignée au chemin SP-1. La demande est alors marquée comme traitée. L'algorithme continue en prenant la demande de trafic de taille juste inférieure tant que toutes les demandes ne sont pas marquées.

- l'algorithme RWA-2, calqué sur l'algorithme RWA-1, se laisse la probabilité de choisir un chemin alternatif SP-2 si le chemin SP-1 ne présente pas de longueurs d'onde accessible. Il est détaillé à la Figure 2.2.

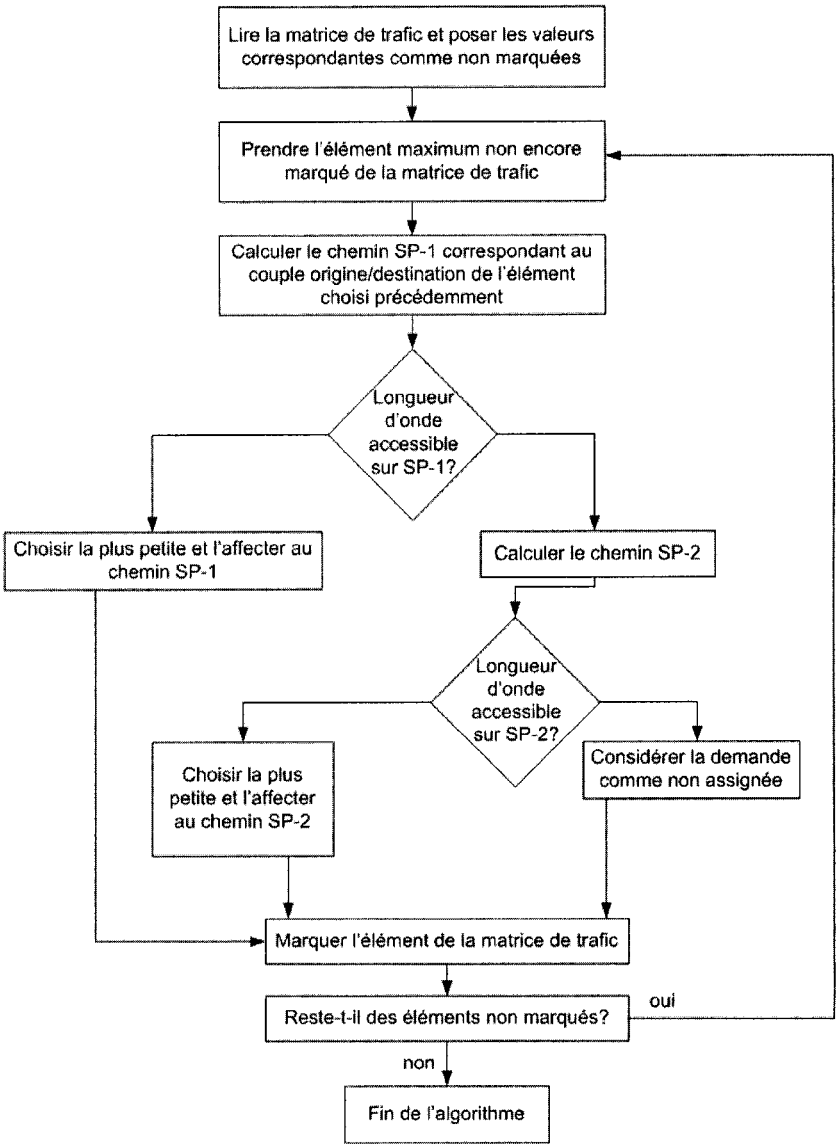



**Figure 2.2** L'algorithme RWA-2

- l'algorithme RWA-3, basé sur l'algorithme RWA2, considère le chemin alternatif SP-3 si les deux premiers chemins SP-1 et SP-2 n'ont pu être assignés à une longueur d'onde.

Les tests sont effectués sur des réseaux et des matrices de trafic générés aléatoirement. La taille des réseaux est fixé à 20 nœuds. Le nombre de longueurs d'onde par fibre (ou lien ici) varie entre 5 et 15. Les résultats obtenus sont consignés dans le Tableau 2.2.

**Tableau 2.2** Résultats comparatifs des algorithmes RWA-1, RWA-2 et RWA-3

| longueurs d'onde | Trafic non assigné (%) | | | Temps d'exécution | | |
|---|---|---|---|---|---|---|
| W | RWA-1 | RWA-2 | RWA-3 | RWA-1 | RWA-2 | RWA-3 |
| 5 | 27,43 | 19,13 | 14,72 | 5,44 | 9,99 | 21,94 |
| 8 | 12,03 | 6,24 | 3,68 | 5,16 | 8,05 | 13,33 |
| 10 | 6,98 | 2,73 | 2,07 | 4,76 | 6,66 | 10,27 |
| 12 | 2,88 | 1,04 | 0,98 | 4,55 | 5,83 | 7,22 |
| 15 | 0,59 | 0,15 | 0,15 | 4,16 | 4,99 | 5,27 |

L'auteur conclue, à la vue de ces résultats, que l'intérêt des chemins alternatif est prédominant pour les fibres possédant un faible nombre de longueurs d'onde (entre 5 et 10 longueurs d'onde), alors qu'il s'estompe quand le nombre de longueurs d'onde par fibre dépasse la dizaine. Il affirme ainsi que l'algorithme RWA-2 est un bon compromis (en comparant les temps de calcul) pour optimiser des réseaux réels, quelque soit le nombre de longueurs d'onde par fibre utilisé.

**Routage dynamique**

Le routage dynamique ne rentre pas exactement dans le cadre de notre étude mais nous avons choisi de présenter rapidement quelques axes de réflexion sur le sujet. Ceci apporte notamment des concepts intéressants sur la gestion d'une liste de plusieurs routes possibles reliant une paire de nœuds du réseau. Une partie de cette liste est souvent générée statiquement alors que l'autre partie est traité dynamiquement. Dans un

routage optique dynamique, une résolution exacte requiert une gestion très complexe de l'information de l'état de tous les liens du réseau. Maintenir et transmettre une telle information nécessite des flots de données énormes et, quand on connaît en plus la fréquence des changements de l'état du réseau, on se rend compte de la difficulté de mettre à jour une information exacte sur la totalité des liens.

Ainsi, les approches classiques tentent de prendre en compte une approximation de la charge du réseau afin de minimiser le risque de congestion. Deux algorithmes de la littérature répondent à cette problématique : l'algorithme de routage de charge minimale (LLR pour Least Load Routing) et l'algorithme de chemins fixés de congestion minimale (FPLC pour Fixed-Path Least-Congestion).

Pour le premier, le critère de sélection d'un chemin, parmi une liste préétablie de différents chemins possibles, est la capacité libre maximale du lien le plus chargé. Dans le second cas, l'algorithme sélectionne la route qui présente le plus grand nombre de chemins optiques disponibles. Ce dernier donne une estimation plus rigoureuse de la congestion dans le réseau, car il prend en compte la contrainte de continuité de longueur d'onde. Des simulations montrent que l'algorithme FPLC apporte un gain de performance supérieur à ses rivaux, mais les plus longs délais de configuration et les plus longs en-têtes de contrôle qu'il engendre le rendent peu efficace. L'algorithme FPLC-k [truc] permet de restreindre l'information de contrôle nécessaire : il prend seulement en compte les $k$ premiers liens de chaque chemin lors du routage. La performance de cet algorithme chute cependant très rapidement lorsque $k$ diminue, et peut même devenir moins bonne que l'algorithme de base du plus court chemin lorsque $k$ est égal à la longueur statistique des chemins du réseau.

L'algorithme de routage adaptatif de charge minimale (ALLR-k pour Adaptative Least Load Routing [28]) s'est inspiré de l'algorithme FPLC-k. L'algorithme LLR est modifié dans le but de déterminer si la mesure de la bande passante libre sur les liens est suffisante pour donner une bonne évaluation de la congestion du réseau optique. L'algorithme proposé se restreint à l'étude séparée du routage et de l'affectation de longueurs d'onde.

Pour ce qui est du routage, l'algorithme ALLR-k se base sur une liste semi-évolutive des chemins possibles entre un nœud source et un nœud destination. Celle-ci comprend d'abord le plus court chemin et une paire de plus courts chemins disjoints (ces trois chemins étant fixés une fois pour toute, ils ne dépendent pas de l'évolution du trafic dans le réseau). Elle comprend également le plus court chemin de largeur de bande maximale et une paire de plus courts chemins disjoints de largeur de bande maximale (ces chemins constituant la partie évolutive de l'algorithme, ils dépendent de l'état d'utilisation des liens). Les plus courts chemins de largeur de bande maximale sont déterminés en attribuant des coûts convexes aux liens (et non plus unitaires comme dans le cas du plus court chemin) qui dépendent de leur utilisation. L'algorithme explore alors jusqu'à $k$ chemins de la liste pour trouver un chemin possible. Il faut bien comprendre ici que l'algorithme LLR n'a qu'une vision déformée de la congestion réelle du réseau (notamment parce qu'il ignore la contrainte de continuité de longueurs d'onde) et que l'algorithme ALLR-k tente d'y remédier en explorant davantage de possibilités dans la liste proposée. La complexité en temps de calcul de cet algorithme est $O(kKHW)$ (où $H$ désigne la longueur maximale de tous les chemins de la liste composée de $K$ plus courts chemins, et $W$ le nombre de longueurs d'onde par fibre optique), ce qui est bien meilleur que celle de l'algorithme FPLC qui est $O(N^2HW^5)$ (où $N$ désigne le nombre de nœuds du réseau).

Pour ce qui est de l'affectation de longueurs d'onde, l'algorithme ALLR-k se déroule en trois étapes : la source émet jusqu'à $k$ paquets selon les $k$ chemins sélectionnés, la destination lance alors un algorithme d'affectation de longueurs d'onde basé sur la longueur d'onde la moins utilisée et renvoie ensuite un paquet au nœud source pour réserver la longueur d'onde choisie.

Les auteurs ont testé les différents algorithmes sur la topologie du réseau NSFNET et sur une topologie aléatoire, lorsque la probabilité d'une nouvelle connexion au réseau suit une loi de Poisson. La performance est évaluée suivant trois critères : la probabilité de blocage (probabilité de refuser une requête de connexion à cause de

l'indisponibilité de chemin), l'utilisation des liens (pourcentage moyen du temps d'utilisation d'une longueur sur un lien) et le rapport fibre-longueur d'onde (rapport entre le nombre de fibres par lien optique et le nombre de longueurs d'onde par fibre). Ce dernier rapport tente de simuler les effets de convertisseurs de longueur d'onde et nous ne détaillerons donc pas les résultats obtenus pour ce critère. Les résultats obtenus sont les suivants :

- LLR-k (la liste n'est pas semi-adaptative pour cet algorithme mais fixe) est plus performant que FPLC-k et plus robustes que ce dernier (au sens où la valeur optimale de $k$ est indépendante de la topologie du réseau);
- LLR-2 et LLR-3 apportent un gain de performance notable par rapport à LLR;
- ALLR-k se comporte aussi bien voire mieux que FPLC.

L'algorithme ALLR-k présente donc des perspectives intéressantes quant à son utilisation pour le routage dynamique dans un réseau optique. Il reste cependant encore à définir exactement les limites précises de sa performance.

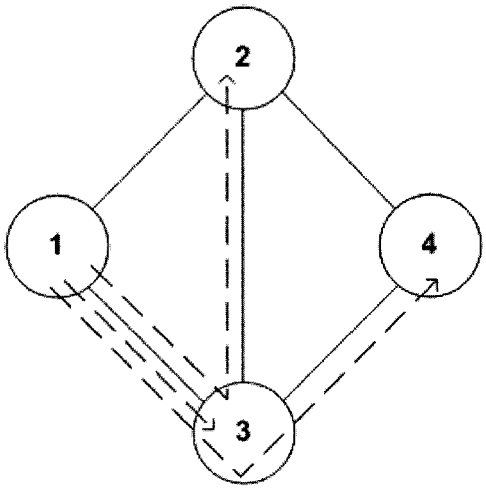
### 2.2.2 Quelques algorithmes d'affectation de longueurs d'onde

Nous présentons ici quelques algorithmes qui se focalisent sur les stratégies d'affectation de longueurs d'onde. Nous nous intéressons à un coloriage classique de graphe dans le cas d'un problème statique.

#### Coloriage de graphe dans le cas statique

Certains chercheurs [22] se sont intéressés au problème RWA dans le cas statique. Ils ont eux aussi scindé le problème complet de routage et d'affectation de longueurs d'onde en deux sous-problèmes traités séparément au moyen d'algorithmes heuristiques. Ils effectuent le routage au moyen d'un algorithme de chemin à plus petit nombre de saut (plus court chemin où les poids des liens sont fixés à un). Les chemins choisis traversent donc un nombre minimal de routeurs optiques, ce qui minimise les délais de transfert aux nœuds intermédiaires.

Par ailleurs, ces auteurs [22] présentent un algorithme d'affectation de longueurs d'onde, l'algorithme du degré de saturation (DSATUR pour Degree of Saturation). L'algorithme DSATUR est reconnu pour le coloriage des sommets d'un graphe [5] et nous décrivons ici une adaptation aux réseaux optiques. À partir du graphe $G_C$ du réseau obtenu à la suite de l'algorithme de routage, il construit un graphe dual $G_P$ (ou graphe des chemins). Nous illustrons un exemple à la Figure 2.3 a) avec un réseau $G_C$ de 4 nœuds. Nous avons représenté en pointillés les chemins optiques issus du routeur 1. La totalité du routage est consignée dans la table de routage correspondante (à la Figure 2.3 b)). La première colonne représente les routeurs sources *s* alors que la première ligne est composée des routeurs destinations *d*. Ainsi, le chemin issu du routeur 1 allant au routeur 4 est rapporté au croisement de la deuxième ligne et la cinquième colonne, et il est constitué de la suite de routeur 1-3-4.




| s \ d | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | 1-3-2 | 1-3 | 1-3-4 |
| 2 | 2-1 | | 2-3 | 2-1-3-4 |
| 3 | 3-1 | 3-1-2 | | 3-2-4 |
| 4 | 4-2-1 | 4-2 | 4-3 | |

a) Graphe $G_C$ b) Table de routage correspondante

**Figure 2.3** Graphe d'un réseau et sa table de routage

Dans $G_P$, chaque nœud est associé à un chemin du graphe $G_C$, et deux nœuds sont reliés si les chemins correspondants possèdent un lien en commun (Figure 2.4).

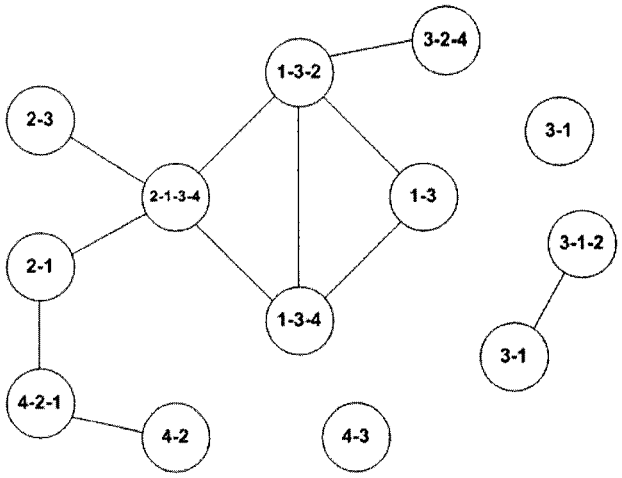



**Figure 2.4** Graphe $G_P$

Le problème d'affectation de longueurs d'onde du graphe $G_C$ devient un problème de coloriage (une couleur correspondant à une longueur d'onde) des nœuds du graphe $G_P$. Le degré de saturation d'un nœud est le nombre de couleurs non permises pour ce nœud (car déjà affectées à un nœud voisin). Les couleurs, ou longueurs d'onde, sont ordonnées aléatoirement. Ce classement n'a pas d'influence (les couleurs jouant des rôles interchangeables) mais il doit rester fixe durant l'exécution de la méthode. L'algorithme DSATUR se décrit alors comme indiqué à la Figure 2.5. Il commence par créer la liste des nœuds de degré de saturation maximal (ceux qui sont les plus contraints parmi les affectations réalisées sur leurs voisins). Il choisit alors dans la liste celui qui a le plus de voisins dans le graphe. Ainsi, à la première itération, aucune affectation n'étant opérée, le nœud choisi est celui qui a le plus de voisins (nœud de degré maximal). Si plusieurs nœuds ont le même degré, l'algorithme en sélectionne un de façon aléatoire. L'algorithme affecte alors à ce nœud la première couleur acceptable, suivant l'ordre précédemment défini. Cette couleur est alors retirée des couleurs permises pour les voisins directs de ce nœud. De même, tous les arêtes incidentes à ce nœud sont ôtées du graphe.

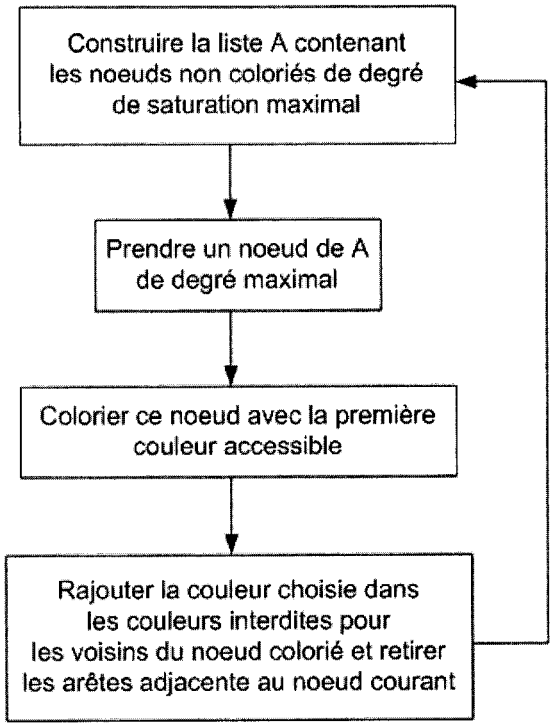



**Figure 2.5** L'algorithme DSATUR

Ainsi, en l'appliquant à l'exemple précédent, les deux nœuds de degré maximal sont les nœuds 2-1-3-4 et 1-3-2. On choisit par exemple le nœud 1-3-2 et on lui affecte la première couleur. Ses quatre voisins voient donc leurs degrés de saturation augmenter d'une unité, et quatre arêtes sont ôtées du graphe pour aboutir au graphe de la Figure 2.6. Seuls les degrés de saturation non nuls sont représentés. À la prochaine itération, le nœud 2-1-3-4 va être choisi car il est dans la liste des nœuds de degré de saturation maximal (égal à 1) et il possède parmi elle le plus grand degré. La deuxième couleur est alors affectée à ce nœud. Les itérations se poursuivent alors jusqu'à ce que tous les nœuds soient coloriés.

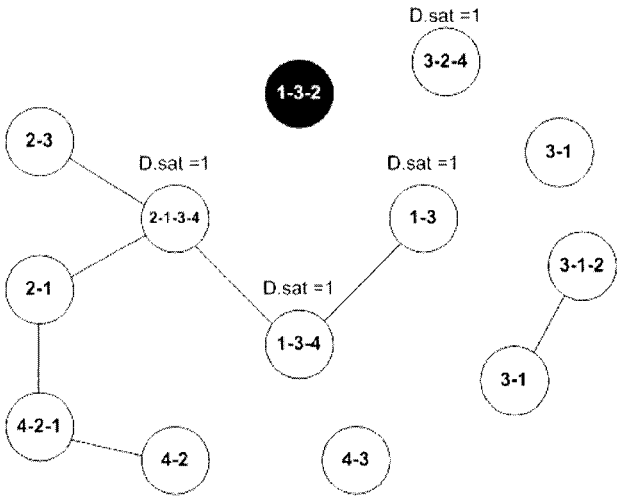



**Figure 2.6** Réseau après la première itération de DSATUR

Des simulations ont été réalisées sur la topologie NSFNET, où la capacité des fibres est fixée à 10 Gb/s par longueur d'onde. Les temps d'exécution des programmes sont de l'ordre de la seconde. Pour une demande uniforme de trafic entre chaque paire de nœuds du réseau (1 Gb/s), l'algorithme trouve qu'il faut 5 longueurs d'onde pour satisfaire la demande. Cependant, une telle demande ne traduit pas fidèlement les matrices de trafic réelles. Ainsi, la matrice de trafic est modifiée : toutes les paires source/destination passant par un certain nœud du réseau (le nœud 5 dans cet exemple) ont une demande de 8 Gb/s au lieu de 1 Gb/s. L'algorithme trouve alors que 11 longueurs d'onde au moins sont nécessaires.

Une telle approche de coloriage pour l'affectation de longueurs d'onde est très fréquente dans la littérature.

Nous avons produit un état de l'art sur le problème d'affectation de longueurs d'onde dans le cas dynamique. Ne rentrant pas exactement dans la problématique exposée dans ce mémoire, nous l'avons consigné en annexe.

### 2.2.3 Étude des routeurs latins

Les routeurs étudiés jusqu'à présent n'imposent aucune contrainte spécifique au réseau. Il existe cependant des routeurs latins (LR pour Latin Router) qui présentent le double avantage d'être peu onéreux et d'avoir une très bonne robustesse (génèrent peu d'erreurs) mais qui ajoutent de nouvelles contraintes au réseau. Des détails de conception sont présentés dans les références [3] et [4]. Nous nous limiterons dans ce mémoire à examiner les contraintes induites par ces routeurs sans expliciter leur origine.

#### Contraintes imposées par les routeurs latins

Chaque routeur latin est composé de ports d'entrée et de ports de sortie. Ces routeurs ont la particularité d'avoir autant de ports d'entrée que de ports de sortie, et ce nombre correspond à la taille du routeur latin. Les routeurs latins sont représentés par une matrice, comme exposé à la Figure 2.7. Cette matrice est de carré latin : on ne rencontre jamais deux fois la même lettre dans une ligne ou dans une colonne. Nous allons expliciter comment cette représentation schématise l'ensemble des contraintes imposées par les routeurs latins.

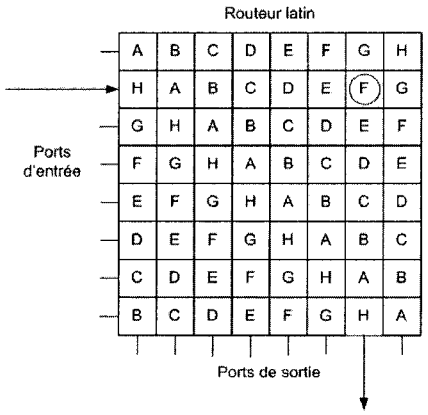



**Figure 2.7** Routeur latin de taille 8

Un chemin optique (ou connexion) entre une source *s* et une destination *d* est initié au niveau de la source, transmis par zéro, un, ou plusieurs routeur(s)

intermédiaire(s) et achevé au niveau de la destination. Le chemin ainsi créé peut servir de canal de communication entre *s* et *d*, mais ne crée aucune connexion entre *s* et les routeurs intermédiaires, pas plus qu'entre *d* et *s* (les connexions sont de bout en bout d'une route et sont orientées).

**Contrainte d'initiation de chemin optique**

Un chemin optique est généré par un laser qui est branché sur un port d'entrée du routeur émetteur, et qui émet sur toutes les longueurs d'ondes disponibles (en nombre égal à la taille du routeur latin choisie). Ce chemin va sortir par un des ports de sortie du routeur latin. Le routeur impose une relation entre le port d'entrée sur lequel est branché le laser, la longueur d'onde que l'on va exploiter pour le chemin que l'on désire construire, et le port de sortie par lequel il va être émis. Cette relation se lit directement sur la matrice du routeur latin : si le laser est branché sur le $i^{ème}$ port d'entrée et que l'on désire émettre sur le port de sortie j, la longueur d'onde utilisé par ce chemin est nécessairement celle qui se trouve à l'intersection de la $i^{ème}$ ligne et de la $j^{ème}$ colonne de la matrice. Les longueurs d'onde sont ici représentées par des lettres. Nous illustrons un exemple d'initiation de chemin optique à la Figure 2.8.

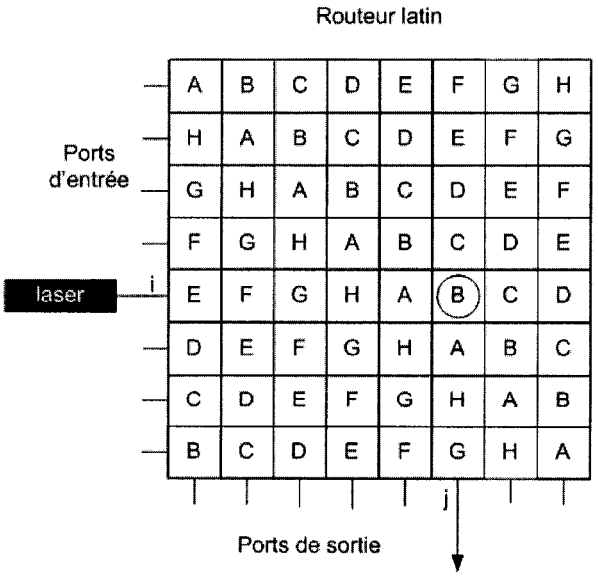



**Figure 2.8** Contrainte d'initiation de chemin optique

Le laser étant branché sur le cinquième port, la longueur d'onde issue du port de sortie six est *B*. De même, pour utiliser F comme longueur d'onde, il faut extraire le chemin optique par le second port de sortie. Afin d'être sûr que chaque routeur latin puisse émettre (donc que tous ses ports d'entrée ne soient pas déjà connectés dans le réseau à des ports de sortie pour pouvoir brancher au moins un laser), le dernier port d'entrée n'est jamais connecté à un port de sortie. Il est nommé port virtuel d'entrée et est noté *S*.

**Contrainte de transition de chemin optique**

Une fois initié, le chemin optique peut transiter par plusieurs routeurs intermédiaires. Le chemin optique arrive sur le routeur latin courant par un port d'entrée (connecté au port de sortie utilisé par le routeur précédent pour ce chemin optique). Le chemin optique a donc un port d'entrée *i* et une longueur d'onde *B* fixés. Les contraintes du routeur latin imposent donc le port de sortie du chemin optique.

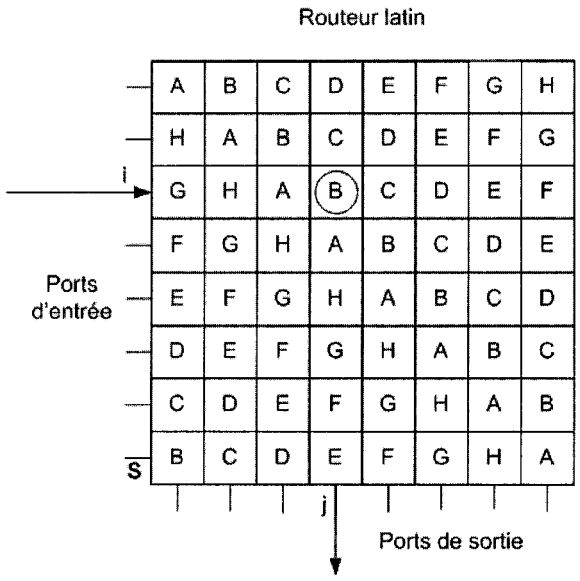



**Figure 2.9** Contrainte de transition de chemin optique

À la Figure 2.9, nous montrons comment un chemin optique arrivant par le troisième port d'entrée à la longueur d'onde B est transmis au routeur suivant sur le chemin optique par le quatrième port de sortie.

**Contrainte de terminaison de chemin optique**

De façon similaire à l'initiation de chemin optique, la terminaison de chemin est réalisée en connectant un récepteur optique sur un port de sortie du routeur latin destination. Une fois le port d'entrée et la longueur d'onde du chemin choisis, le port de sortie est déterminé par les contraintes schématisées par la matrice de carré latin. Un exemple est présenté à la Figure 2.10.

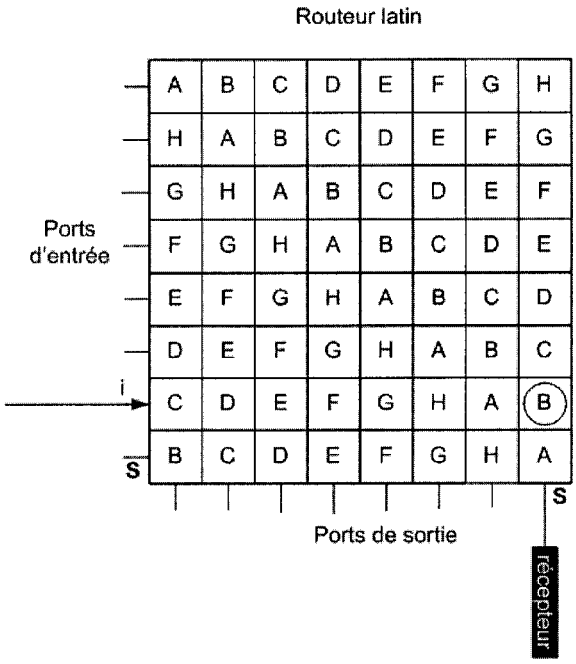



**Figure 2.10** Contrainte de terminaison de chemin optique

À nouveau, pour s'assurer de la possibilité pour chaque routeur d'être récepteur, le dernier port de sortie est un port virtuel et n'est jamais connecté. Il est encore noté *S*. Nous récapitulons à la Figure 2.11 les différentes composantes d'un chemin optique :

- émission d'un signal sur un port de sortie du routeur source;
- propagation du signal, à longueur d'onde constante, à travers les routeurs intermédiaires en respectant les contraintes de chaque routeur (relations entre port d'entrée et de sortie);
- réception du signal par le routeur destination.

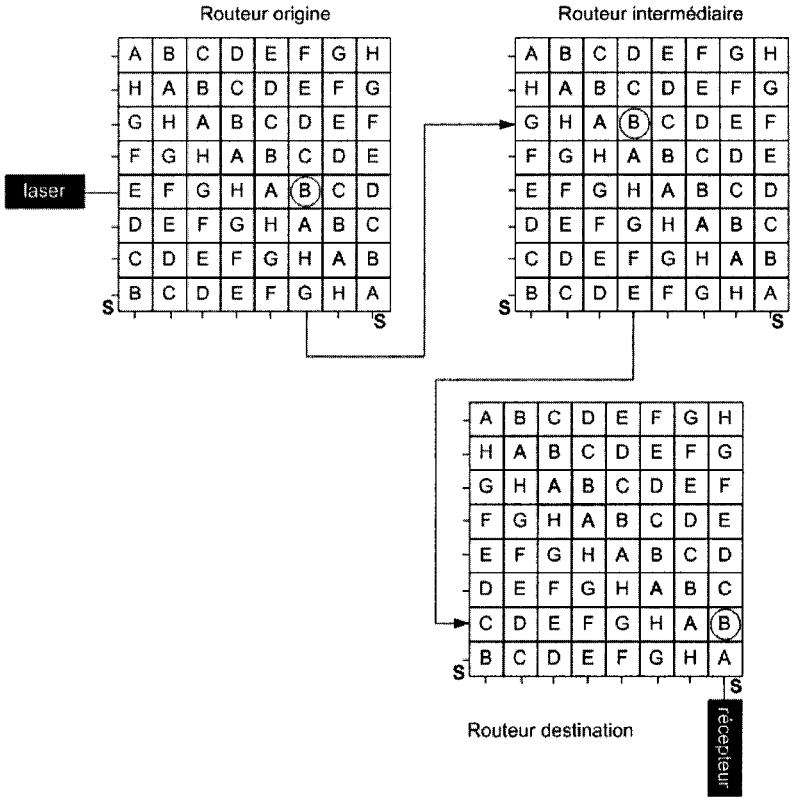



**Figure 2.11** Contraintes imposées par les routeurs latins

Nous avons généré à la Figure 2.12 un exemple simplifié de routage avec un routeur 3x3, pour mettre en valeur les conséquences de l'utilisation de routeurs latins.

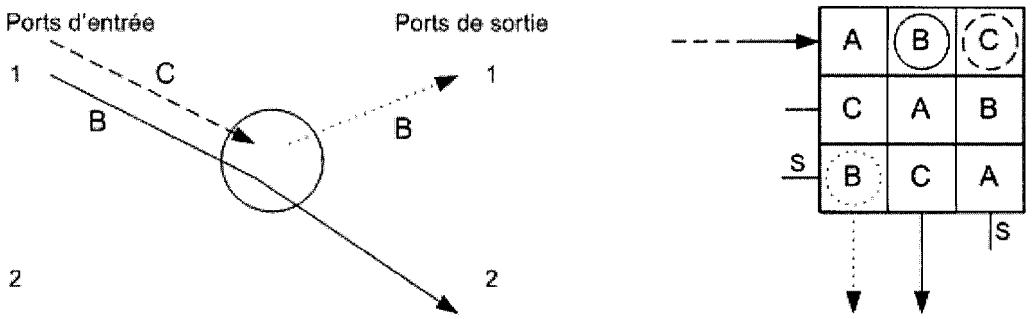



**Figure 2.12** Routeur de carré latin de taille 3

Ces nouvelles contraintes rendent difficile la séparation entre routage et affectation de longueurs d'onde. En effet, considérons le routage préétabli sur la gauche de la Figure 2.13. À ce stade du routage, la requête de transmission du routeur 3 au routeur 2 ne peut être établie.

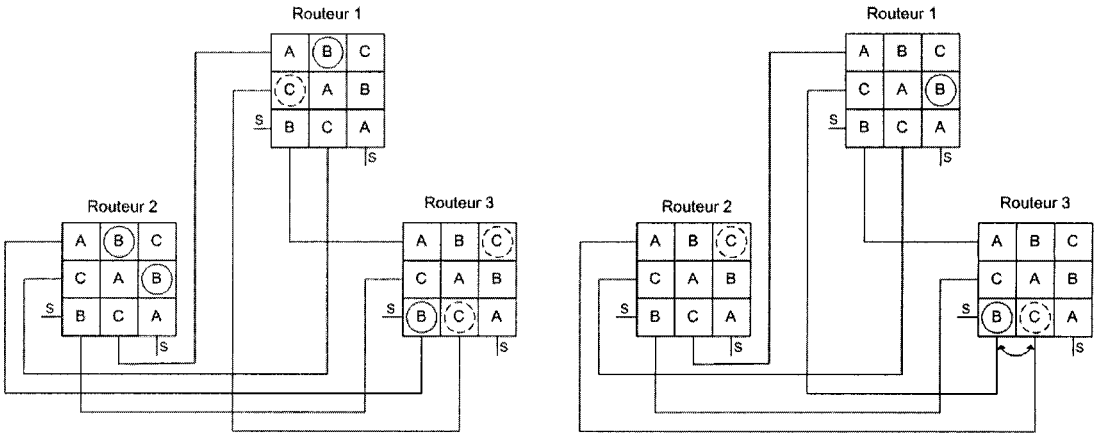
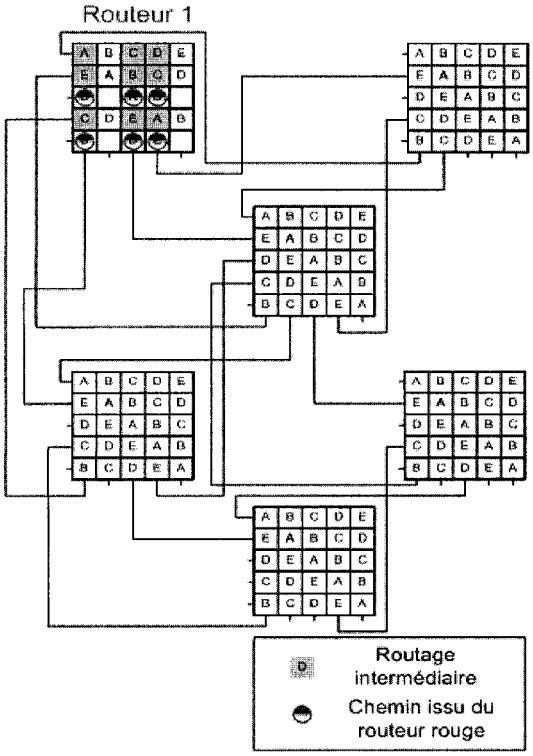



**Figure 2.13** Interversion de ports de sortie

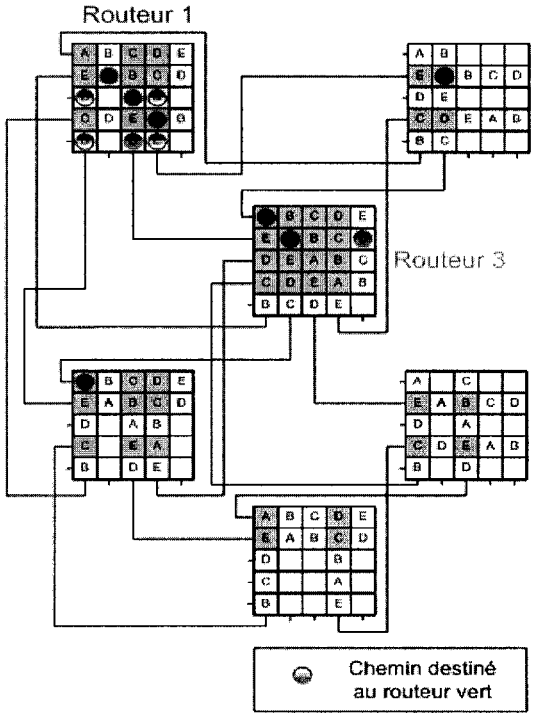
En permutant les connexions des ports de sortie du routeur 3, nous remarquons (schéma de droite de la Figure 2.8) que la requête peut alors être satisfaite. Sur ce petit exemple, nous prenons conscience de la relation étroite entre routage et affectation de longueurs d'onde dans de tels réseaux.

Nous présentons maintenant un exemple plus concret pour illustrer les réels problèmes qui se posent lorsque l'on s'intéresse à optimiser le routage en présence de routeurs latins. Pour ce faire, nous allons expliciter pas à pas toutes les connexions engendrées, dans un réseau de six routeurs 8x8, par le branchement de la Figure 2.14 a). Les différents routeurs ont entre deux et quatre voisins, quatre étant le maximum si l'on considère que le dernier port est laissé libre pour permettre l'émission ou la réception de signal optique.

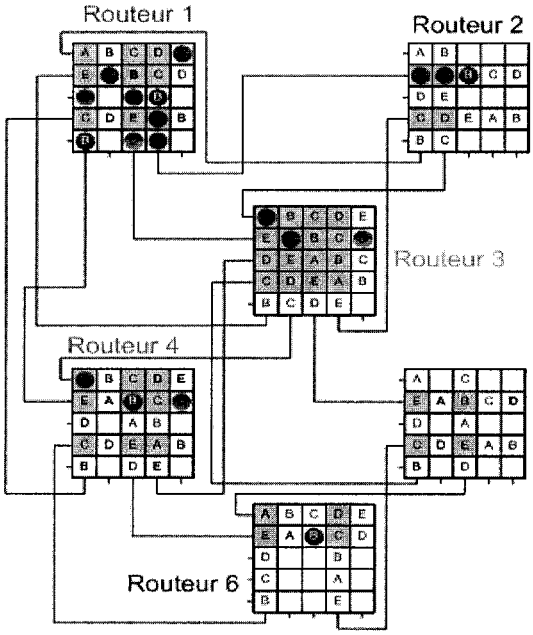
Nous nous intéressons tout d'abord aux connexions issues du routeur 1 (Figure 2.14 a). Les cases grisées du routeur 1 illustrent une intersection de deux ports connectés. À la longueur d'onde correspondante, le routeur sera un intermédiaire sur le chemin optique concerné. Les cases blanches concernent alors une extrémité de chemin optique, à la longueur d'onde associée. Tentons ainsi de repérer les connexions créées par le port de sortie 3 du routeur 1. Ce port est relié au port d'entrée 3 du routeur 3. À la Figure 2.14 b, nous répercutons le traitement effectué au routeur 1 à tous les autres routeurs. Une connexion dans le réseau est schématisée par un rond, dont la couleur du demi-cercle supérieur indique le routeur source et celle du demi-cercle inférieur le routeur destination. La longueur d'onde $D$ produit ainsi une connexion directe entre le routeur 1 et le routeur 3 (couleurs rouge et vert). Pour sa part, la longueur d'onde $A$ génère une connexion entre le routeur 1 et lui-même, après avoir traversé successivement les routeurs 3, 4, 1 et 2. La Figure 2.14 c applique cette méthode à tous les ports de sortie du routeur 1. Nous réalisons finalement les connexions pour l'ensemble des routeurs du réseau (Figure 2.14 d)). Cet exemple est reproduit ici pour mettre en lumière la difficulté introduite par les routeurs latins dans le cadre de l'optimisation de réseaux.



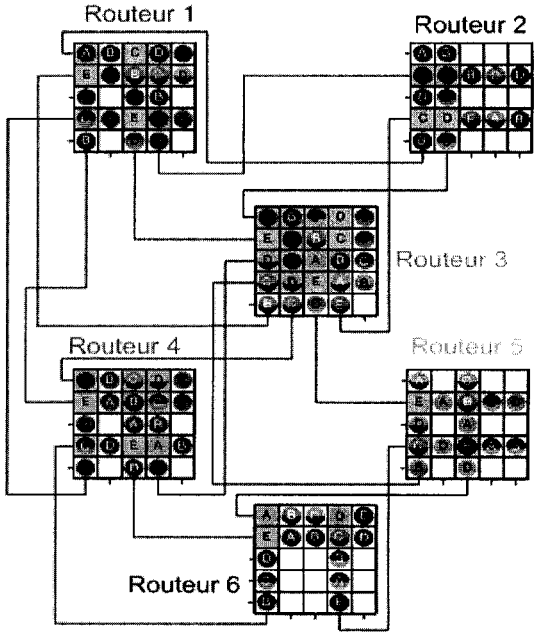

a) Départ des chemins issus du routeur 1




b) Connexions issues du port de sortie 3




c) Connexions issues du routeur 1




d) Connexions totales

**Figure 2.14** Établissement pas à pas des connexions d'un petit réseau

**Exemple d'optimisation de réseaux possédant des routeurs latins : l'algorithme LONCA**

Devant la complexité des problèmes rencontrés ici, des méthodes basées sur la recherche locale (de type permutation de connexions entre ports d'un même routeur) semblent constituer un axe d'étude très intéressant. Banerjee et Frank [1] et [2] proposent ainsi un algorithme de recherche locale pour résoudre le problème RWA dans le cas de routeurs latins : l'algorithme de recherche locale de configuration dans un réseau optique (LONCA pour Local-search Optical Network Configuration Algorithm).

La recherche locale effectue de petits changements sur une affectation complète (solution courante) dans un problème contraint de façon à améliorer cette solution. Cette recherche se place dans le cas de requêtes statiques, et s'impose deux objectifs :

- établir le nombre maximal de connexions dans un réseau de taille fixée ;
- satisfaire le plus grand nombre de connexions parmi un ensemble donné de requêtes.

L'algorithme se déplace localement dans l'espace des topologies virtuelles en interchangeant, pour un routeur du réseau pris au hasard, l'allocation de deux entrées (ou de deux sorties). Cela correspond aux changements de deux lignes (ou de deux colonnes) dans la matrice du routeur. La résolution du routage et de l'affectation de longueurs d'onde se fait ainsi de manière conjointe. La Figure 2.15 montre le fonctionnement de l'algorithme LONCA. Une itération de l'algorithme correspond à choisir aléatoirement un routeur du réseau, à une évaluer le nombre de connexions après chaque interversion de ports de ce routeur (en entrée ou en sortie), à réaliser la meilleure d'entre elles (même si elle n'améliore pas le nombre de connexions satisfaites) et à tester si le nombre maximal d'itérations ou de connexions requises est atteint. L'algorithme relance l'itération suivante si le dernier test s'avère négatif et s'arrête dans le cas contraire.

Les simulations effectuées se basent sur des réseaux aléatoires possédant un degré moyen de 4.5 $N$ (ce qui correspond à la valeur des réseaux optiques actuels). Les routeurs sont de taille $K$=8 dans le cas d'un graphe simple (tous les arcs sont de

multiplicité 1) et de taille $K$=15 lorsque les arcs du graphe sont doubles. Le nombre maximal d'itérations de la recherche locale est fixé à 500. Le nombre de nœuds dans les réseaux générés varie entre 50 et 100. Chaque configuration est testée sur 500 topologies présentant les mêmes caractéristiques.

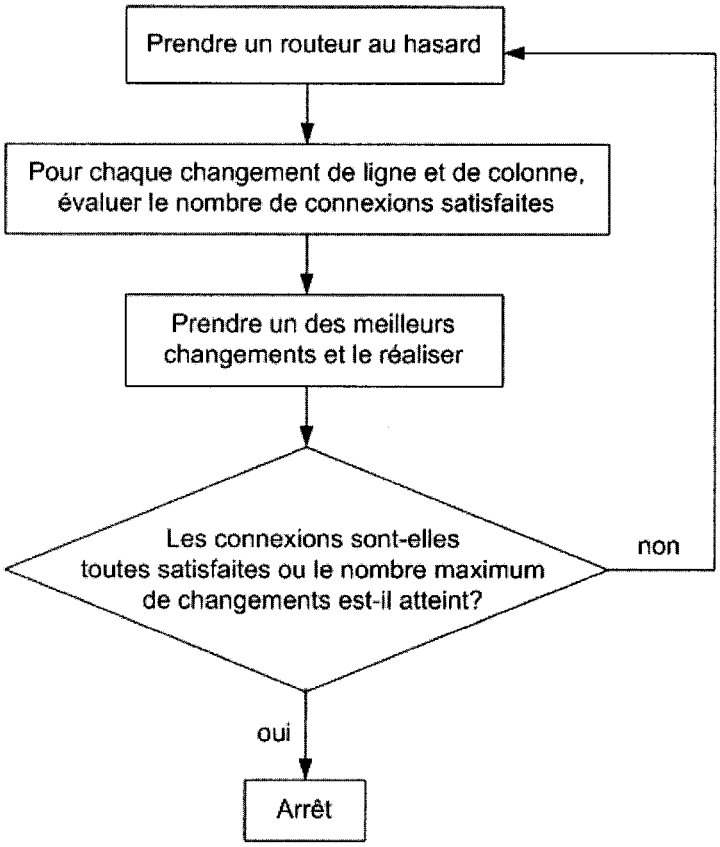



**Figure 2.15** L'algorithme LONCA

Les tests [2] portent tout d'abord sur l'aptitude de l'algorithme LONCA à établir le nombre maximal de connexions dans un réseau de taille fixée. Ils sont présentés au Tableau 2.3 et seront rappelés plus avant dans ce mémoire comme éléments de comparaison. Nous avons sélectionné les résultats concernant l'étude des graphes simples ($K$=1) car notre étude se limitera à analyser ce cas-ci. Le nombre d'itérations de

l'algorithme a été fixé à 500 car les auteurs ont remarqué que les itérations suivantes n'optimisaient plus le routage du réseau.

**Tableau 2.3** Nombre maximal de connections établies par l'algorithme LONCA

| Nombre de nœuds | Moyenne (connexions) | Ecart type (connexions) | Min (connexions) | Max (connexions) | Médiane (connexions) |
|---|---|---|---|---|---|
| N=50 | 564,76 | 17,17 | 511 | 616 | 565 |
| N=60 | 681,87 | 18,14 | 631 | 742 | 683 |
| N=70 | 799,70 | 20,32 | 744 | 863 | 799 |
| N=80 | 914,24 | 21,51 | 855 | 982 | 915 |
| N=90 | 1027,33 | 21,91 | 968 | 1087 | 1035 |
| N=100 | 1144,54 | 24,82 | 1067 | 1230 | 1146 |

L'algorithme est alors testé avec des requêtes précises entre certaines paires de nœuds (matrice de trafic). La taille du réseau est ici fixée à 50 nœuds. Le nombre de requêtes varie entre 100 et 500. Elles sont générées aléatoirement entre plusieurs paires de nœuds du réseau. Les tests sont réalisés sur 5 topologies de réseaux différentes L'algorithme est lancé 100 fois pour chaque topologie. Le tableau 2.4 présente les résultats obtenus.

**Tableau 2.4** Nombre de connections satisfaites suivant le nombre de requêtes

| Demande (connexions) | Topologie 1 (connexions) | Topologie 2 (connexions) | Topologie 3 (connexions) | Topologie 4 (connexions) | Topologie 5 (connexions) |
|---|---|---|---|---|---|
| 100 | 63,25 | 66,29 | 62,09 | 62,29 | 62,43 |
| 150 | 84,37 | 84,36 | 84,07 | 85,72 | 83,06 |
| 200 | 103,37 | 101,34 | 102,52 | 105,82 | 104,34 |
| 250 | 120,42 | 118,93 | 119,04 | 121,63 | 121,66 |
| 300 | 136,17 | 140,72 | 137,08 | 139,71 | 138,07 |
| 350 | 152,03 | 154,76 | 151,12 | 157,64 | 153,01 |
| 400 | 165,47 | 170,12 | 166,18 | 171,67 | 170,67 |
| 450 | 179,40 | 187,24 | 179,49 | 186,89 | 185,40 |
| 500 | 191,72 | 202,38 | 190,05 | 201,85 | 202,04 |

Cette étude est la première recherche, à ma connaissance, à fournir des résultats précis sur une solution générale au problème RWA pour des réseaux contenant des routeurs latins. Elle constituera notre référence pour estimer les performances de notre algorithme développé dans ce mémoire.

Nous présentons maintenant l'état de l'art sur les méthodes que nous allons employer dans la conception de notre algorithme.

## 2.3 Recherche locale et recherche à voisinage variable

Nous exposons tout d'abord les principaux concepts de la recherche locale.

### Recherche locale

Pour de nombreux problèmes d'optimisation combinatoire, les algorithmes exacts qui garantissent la complétude de la solution ne sont efficaces que dans le cas de problèmes de taille raisonnable. Au-delà, la complexité des problèmes mis en jeu rendent ces algorithmes peu performants. La recherche locale est une alternative très connue à ces méthodes exactes. Cette heuristique (qui perd en complétude mais gagne en efficacité) est souvent utilisée pour tenter de résoudre de tels problèmes. À partir d'une solution initiale du problème (aléatoire ou non optimale), elle tente d'explorer (exhaustivement ou pas) un voisinage de cette solution pour migrer vers un voisin plus attractif. Ce processus est alors répété itérativement à partir de la nouvelle solution courante. Les deux grands concepts qui régissent cette méthode sont la définition du voisinage et le choix du voisin parmi ce voisinage. Une représentation générale de la méthode de recherche locale est proposée à la Figure 2.16.

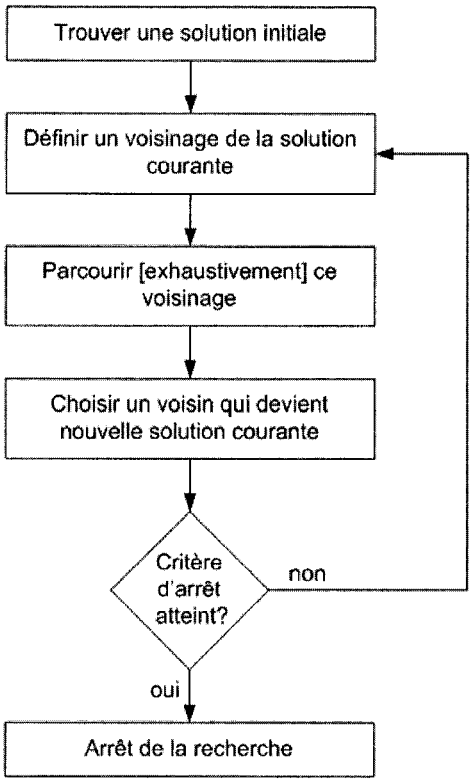



**Figure 2.16** Algorithme général de recherche locale

Cette méthode est cependant très sensible aux optima locaux. En effet, n'ayant qu'une vision partielle de l'espace des solutions, elle peut se retrouver confinée au sein d'un voisinage dans lequel elle n'aperçoit pas de meilleure solution que la solution courante, bien qu'un optimum global puisse exister ailleurs. Ce cas est illustré à la Figure 2.17 où la recherche locale de descente simple (voisinage fixe, choix du meilleur des voisins à chaque itération) s'arrête sur un minimum local.

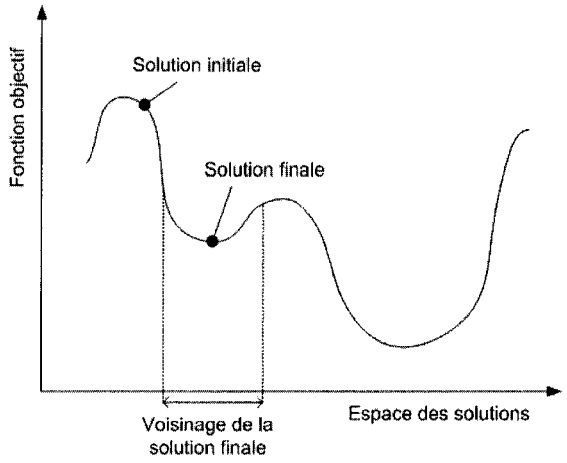



**Figure 2.17** Méthode de descente simple

Plusieurs méthodes ont été développées pour que la recherche locale puisse s'échapper des optima locaux. Elles sont basées sur trois différentes stratégies :

- choisir de bons voisins ;
- définir de grands voisinages ;
- conserver de l'information sur une grande partie de l'espace des solutions.

Les méthodes qui se focalisent sur l'optimisation du choix du voisin sont dites méthodes de dégradation. Elles acceptent généralement de pouvoir dégrader temporairement la solution courante pour aboutir dans des zones de l'espace des solutions inexplorées et intéressantes. Nous pouvons citer parmi elles la recherche taboue (qui maintient une liste de mouvements interdits obligeant parfois la recherche locale à effectuer des mouvements à priori non optima, proposée par Glover en 1989-1990 [7] et [8] ) et le recuit simulé (qui confère un probabilité de réalisation non nulle à un mouvement qui dégraderait la fonction objectif, proposé par Kirkpatrick en 1983 [13] ).

La seconde stratégie consiste à définir des voisinages les plus grands possibles. La recherche locale se doit alors de trouver des métaheuristiques pour parcourir rapidement ces grands voisinages. Nous pouvons citer par exemple l'utilisation de la programmation par contrainte pour guider la recherche dans les grands voisinages.

Enfin, la troisième stratégie vise à ne pas se borner à une seule vision locale pour guider le choix des voisins de la solution courante. Nous pouvons citer la recherche locale à plusieurs solutions initiales (qui consiste à opérer une descente simple sur plusieurs solutions initiales distinctes) ou la recherche locale associée à des algorithmes génétiques (qui consistent à combiner plusieurs solutions courantes, appelées gènes, au moyen de recouvrements, de mutations et de sélections pour obtenir une descendance optimisée). Ces derniers ont été imaginés par Holland en 1962 [11] et vulgarisés par Goldberg en 1989 [9].

Nous allons détailler ici une méthode de recherche locale à voisinage variable, qui joue sur une liste de différents voisinages pour chaque solution courante afin d'échapper aux optima locaux. Elle pourrait ainsi être cataloguée dans la deuxième famille des stratégies présentées ci-dessus.

**Recherche à voisinage variable : méthode VNS**

Cette métaheuristique (VNS pour Variable Neighbourhood Search) a été proposée par Mladenovic en 1995 [17]. Nous détaillons ici la version présentée en [10] par Hansen et Mladenovic. Leur méthode se propose d'essayer une succession de plusieurs voisinages si le premier d'entre eux n'a pas fourni de voisin optimisant la fonction objectif. Le schéma général est reproduit à la Figure 2.18. La première phase est une phase d'initialisation où l'on choisit un ensemble fini $N_k$ $(k = 1,\ldots,k_{\max})$ de voisinages présélectionnées, où l'on détermine un critère d'arrêt et où l'on trouve une solution initiale. Ensuite, la méthode sonde les voisinages successifs de $N_k$ du point courant tant qu'elle ne trouve pas de voisin satisfaisant. Cette recherche locale au niveau de chaque voisinage se fait en trois temps :

- perturbation en générant un voisin aléatoire de la solution courante ;
- recherche locale sur ce voisin, en utilisant le voisinage courant, pour exhiber l'optimum local ;

- décision d'effectuer ou non le mouvement. Si le mouvement est jugé efficace, le voisin choisi devient la nouvelle solution courante et la méthode est relancé sur le premier voisinage. Dans le cas contraire, la solution courante est inchangée et le voisinage suivant est testé.

La méthode s'achève une fois le critère d'arrêt atteint.

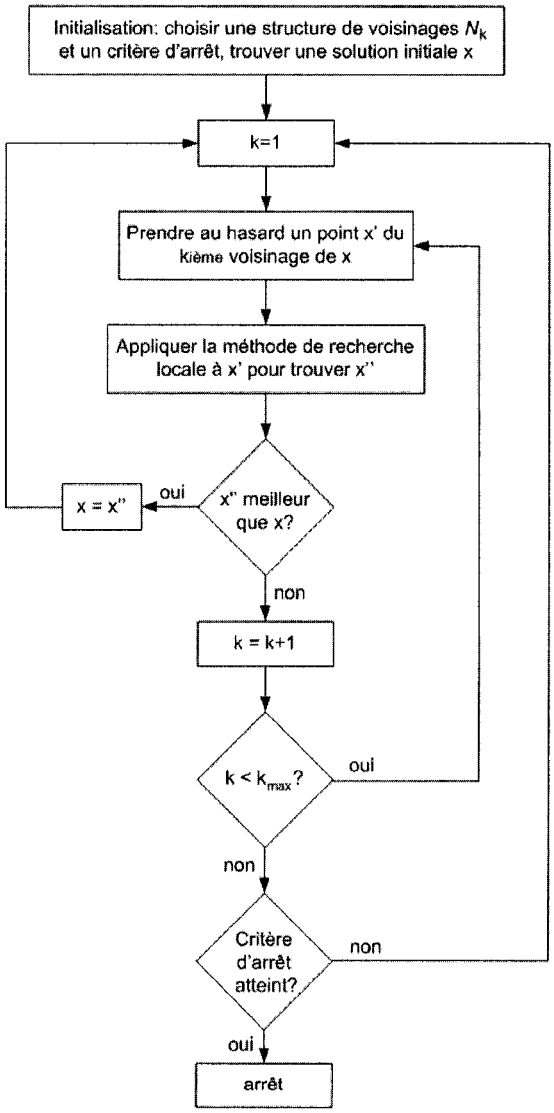



**Figure 2.18** Méthode VNS

La recherche à voisinage variable a été testée [17] sur de nombreux problèmes connus d'optimisation combinatoire et même d'optimisation globale. Les résultats fournis

(nottament pour les problèmes suivants : problème du voyageur de commerce, problème du p médian, problème de Weber à sources multiples) sont probants et confirment l'intérêt actuel de la recherche sur cette métaheuristique.

Nous présentons dans le chapitre suivant une adaptation d'une méthode de recherche locale à voisinage variable pour notre problème de routage optique dans des réseaux comportant des routeurs latins.

# CHAPITRE III
# ALGORITHME VNSFOR PROPOSÉ POUR LE ROUTAGE OPTIQUE

Ce chapitre expose l'adaptation de la méthode de recherche locale à voisinage variable pour la résolution du problème de routage optique en présence de routeurs latins. Notre but initial est de construire une heuristique qui pourra répondre à deux objectifs distincts : établir le maximum de connexions à l'intérieur du réseau ou répondre à une demande particulière de connexions en tentant d'en satisfaire le plus possible. Nous définissons tout d'abord la problématique exacte abordée dans ce mémoire. Nous détaillons ensuite les choix que nous avons effectués pour l'adaptation de la méthode VNS (choix d'une solution initiale, choix d'une structure de voisinages, choix d'un critère d'arrêt, définition de la recherche locale). Nous finirons ce présent chapitre sur une étude des réseaux à générer pour la validation de notre algorithme.

## 3.1 Présentation de la problématique

Nous basons notre recherche sur des réseaux optiques constitués de routeurs latins identiques (routeurs passifs engendrant des contraintes de routage en plus de la contrainte de continuité de longueur d'onde). Nous supposons fixée la topologie physique du réseau, c'est-à-dire que nous étudions un réseau dont le nombre et la position des routeurs, le nombre, la répartition et les caractéristiques des liens, ont été préalablement définis. L'optimisation consiste alors à déterminer les ports de sortie et les ports d'entrée de chaque lien reliant deux routeurs latins du réseau. Notre travail s'inscrit donc bien comme une partie essentielle d'un problème général bien plus complexe.

Nous considérerons deux scénarii possibles pour l'optimisation : vouloir connecter le plus de nœuds dans le réseau (scénario 1) ou répondre à une demande entre ces nœuds (scénario 2).

Nous exposons tout d'abord les paramètres définissant le problème :

- réseau de $N$ routeurs optiques comprenant $M$ ports d'entrée et $M$ ports de sortie;
- $K$ liens bidirectionnels entre ces routeurs, représentés par la matrice *Liens(i,j)*, où *Liens(i,j)*=1 si $i$ et $j$ sont liés, 0 sinon.

Nous pouvons donc donner la formulation de notre fonction à optimiser suivant les deux scénarii possibles :

- maximiser le nombre de connexions

$$Max\left(\sum_{(i,j)\in[1,N]^2} Connexion(i,j)\right) \qquad \text{(scénario 1)}$$

- répondre à une demande

$$Max\left(\sum_{(i,j)\in[1,N]^2} Demande(i,j)*Connexion(i,j))\right) \qquad \text{(scénario 2)}$$

où la matrice *Connexion(i,j)* représente les connexions engendrées par le routage effectué (*Connexion(i,j)*=1 s'il existe au moins une connexion ayant $i$ comme origine et $j$ comme destination) et avec la matrice de demande de trafic *Demande(i,j)*, pour le scénario 2 (*Demande(i,j)*=1 si l'on désire établir une connexion ayant $i$ comme origine et $j$ comme destination, *0* sinon).

Nous présentons alors l'adaptation de la méthode de recherche locale à voisinage variable désirant répondre à cette problématique. Nous définissons tout d'abord la structure de voisinages de notre algorithme. Nous proposons ensuite une méthode incrémentielle de re-routage nous permettant d'explorer des voisinages de taille importante.

### 3.2 L'algorithme VNSFOR : structure de voisinages

Une fois la recherche locale piégée dans un optimum local (pas d'amélioration au cours d'une itération), nous nous proposons de modifier la structure de notre voisinage pour explorer une nouvelle zone voisine (avec un nouveau mouvement) autour de la solution courante et ainsi tenter d'améliorer encore notre solution. La même méthodologie peut correspondre aisément aux deux scénarii envisagés dans notre mémoire : il suffit d'appliquer une fonction de coût adéquate pour juger l'amélioration de la solution courante. L'algorithme proposé se formalise à la Figure 3.1.

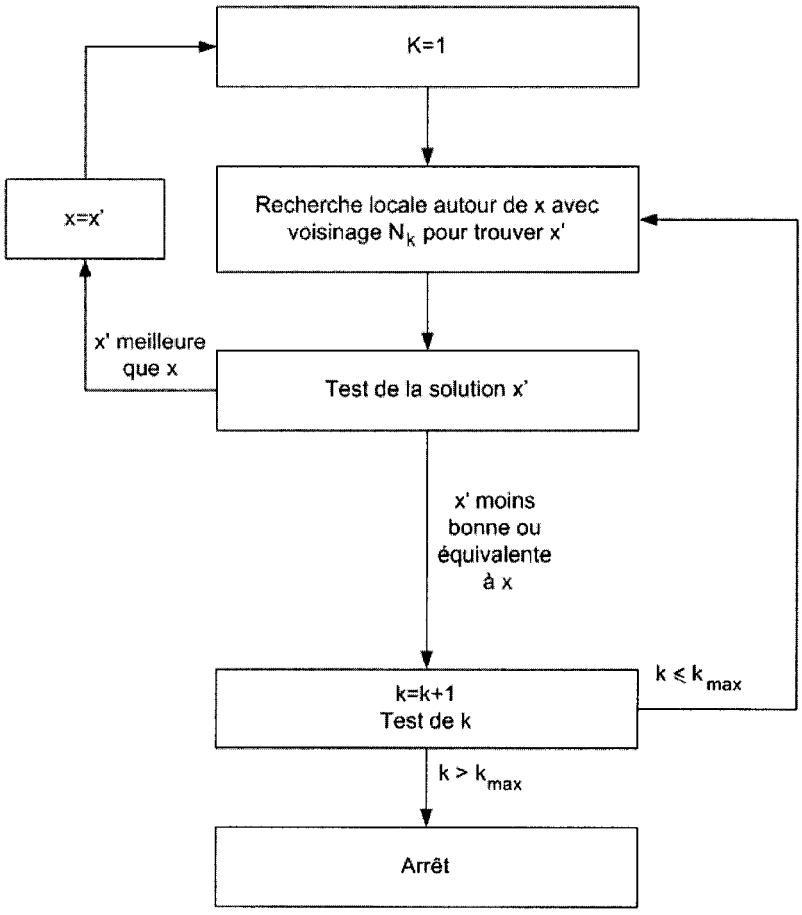



**Figure 3.1** Algorithme de recherche à voisinage variable

Il faut maintenant définir les différents voisinages $N_k$. Nous avons ordonné les mouvements depuis celui qui nous paraissait le plus naïf, jusqu'à celui qui propose le

plus de complexité. L'algorithme teste ainsi, de manière rapide, des mouvements très simples et se laisse la possibilité d'effectuer ensuite des mouvements de plus en plus complexes s'il est pris dans un optimum local.

### 3.2.1 Échange entre ports de sortie : voisinage $N_1$

Le premier mouvement est celui qui est le plus intuitif selon notre modélisation du problème. Il s'agit de la permutation de la connexion de deux ports de sortie d'un routeur pris au hasard. En effet, nos variables de décision (*cx_routeur* et *cx_port*) sont reliées directement aux ports de sortie des routeurs, et ce mouvement s'implémente de manière naturelle. Selon les configurations des ports de sortie, il se représente sous les deux formes de la Figure 3.2.

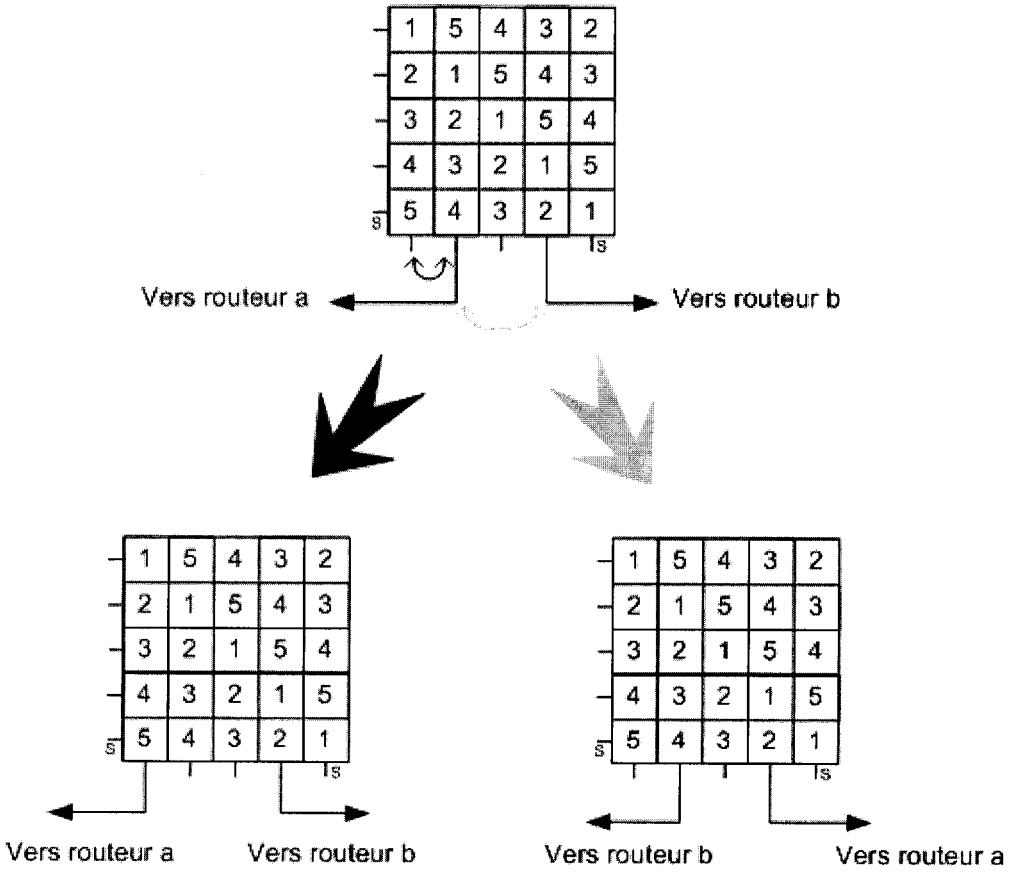



**Figure 3.2** Premier voisinage : échange simple en sortie

L'échange peut s'effectuer entre un port connecté et un port non connecté (flèche noire) ou deux ports connectés (flèche grise). Le voisinage décrit par ce mouvement est alors composé de toutes les feuilles de l'arbre de recherche correspondant au choix d'un routeur dans le réseau, et d'une combinaison de deux de ses ports de sortie (dont au moins un est connecté). Comme vu précédemment dans ce chapitre, la taille de ce voisinage est au maximum de $21*N$ (avec $M$ pris égal à huit). Le maximum traduit ici le pire cas, obtenu lorsque tous les ports sont connectés. Le meilleur cas (pour des réseaux dont tous les nœuds sont au moins de degré deux, donc ont au moins deux voisins directs) est de $11*N$ (cas où seulement deux ports sont connectés).

### 3.2.2 Échange entre ports d'entrée : voisinage $N_2$

Le second voisinage proposé réalise une permutation similaire avec les ports d'entrée d'un routeur. Il est exposé à la Figure 3.3.

Nous échangeons soit un port connecté avec un port non connecté (flèche noire), soit deux ports connectés (flèche grise). La taille du voisinage défini alors est au maximum (si tous les ports possibles sont connectés) de $21*N$ ($M$ étant encore pris égal à huit).

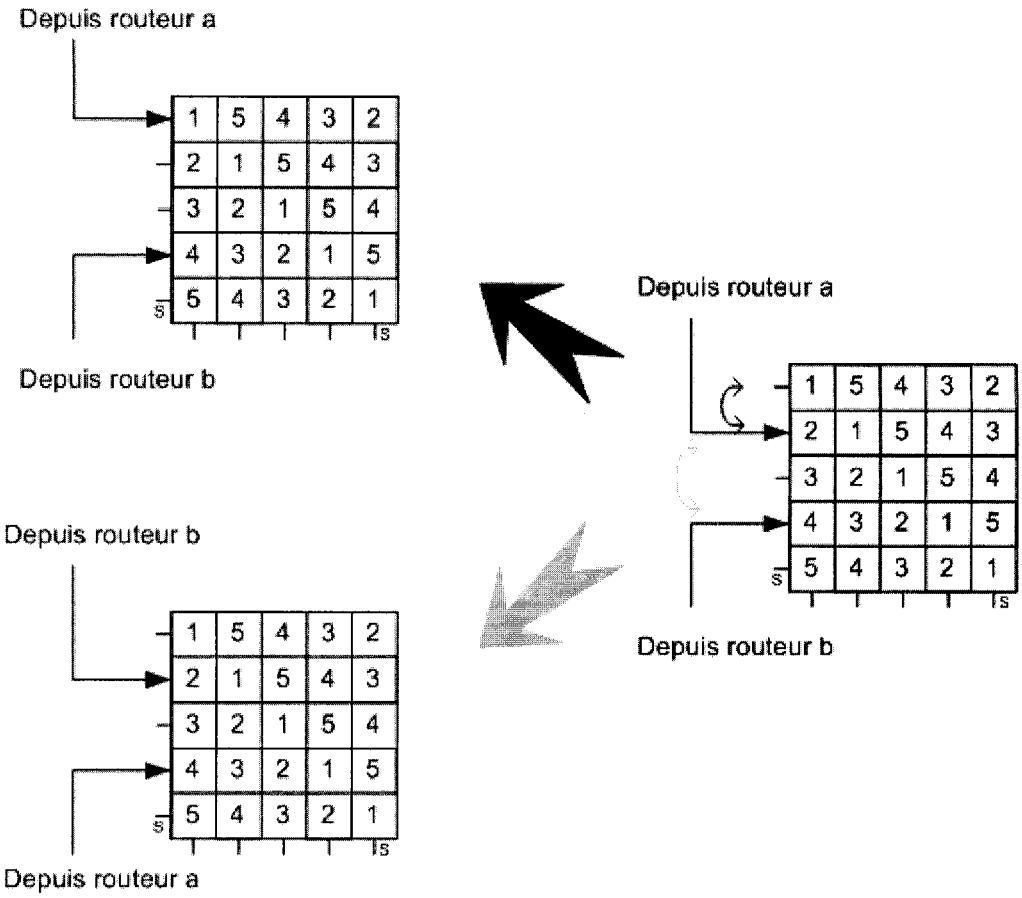



**Figure 3.3** Deuxième voisinage : échange simple en entrée

### 3.2.3 Échange double entre ports d'entrée et ports de sortie : voisinage $N_3$

Pour échapper de manière efficace aux optima locaux, il faut pouvoir être capable d'effectuer des mouvements d'une ampleur bien supérieure aux mouvements définis ci-avant. L'idée d'un mouvement double, combinant les deux précédents, devient alors naturelle face à notre problème étudié. Le mouvement double est décrit à la Figure 3.4.

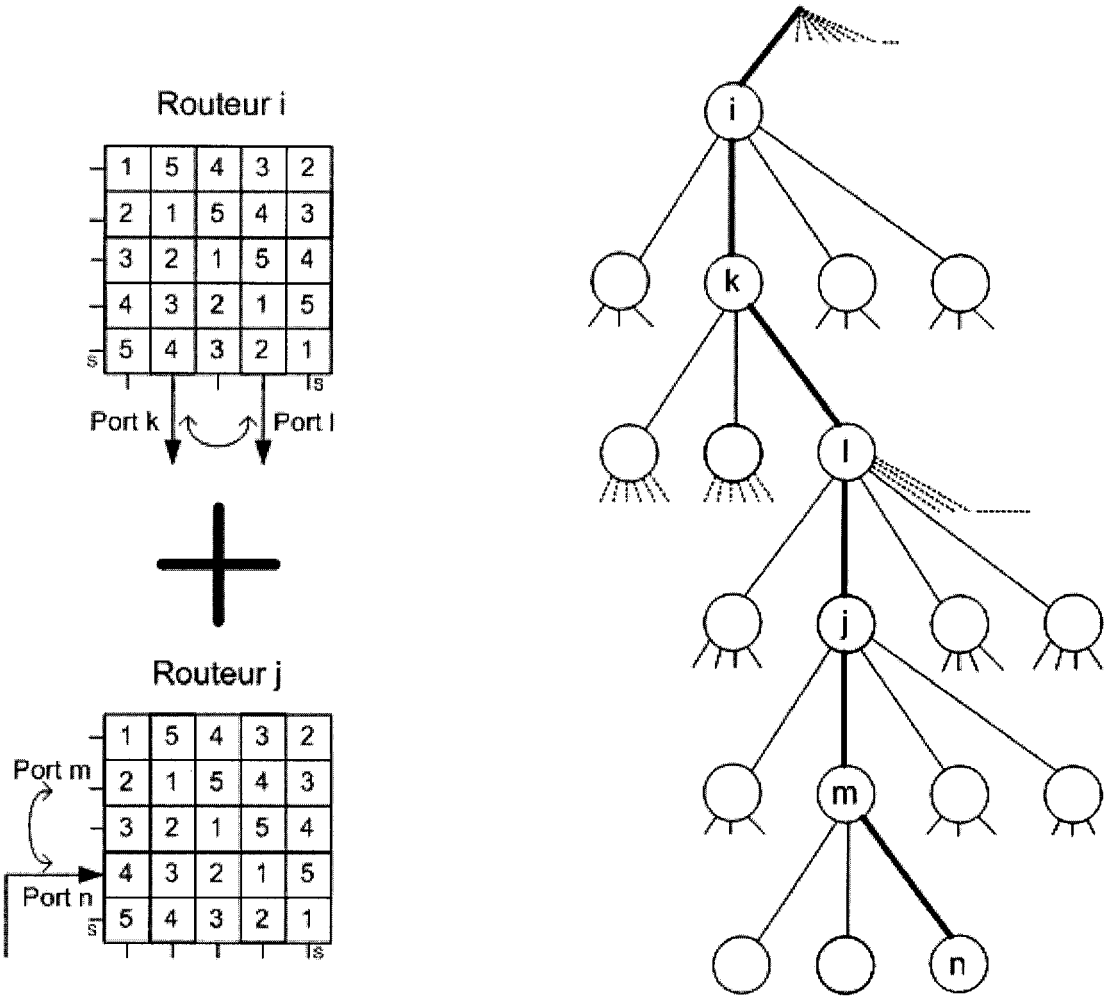



**Figure 3.4** Troisième voisinage (échange en entrée et en sortie) et arbre de recherche

Il est composé d'une permutation entre ports d'entrée et d'une permutation entre ports de sortie. Les routeurs *i* et *j* peuvent éventuellement se retrouver être le même routeur. Les échanges se font à nouveau entre un port connecté et un port non connecté, ou entre deux ports connectés. Cependant, l'arbre de recherche correspondant à ce voisinage (choix indépendant de deux ensembles de trois variables : routeur *i* avec ses ports de sortie *k* et *l*, et routeur *j* avec ses ports *m* et *n*) est gigantesque. Le nombre de feuilles est au maximum de (21**N*)*(21**N*), soit environ 460 $N^2$. Ici, il convient de faire une meilleure approximation du nombre de feuilles en calculant la taille de l'arbre pour

un réseau type, car les ports ne seront jamais tous connectés. Le nombre de voisins d'un routeur est entre deux et sept si l'on exclue les cas des réseaux où certains nœuds sont de degré un. En considérant que chaque degré est équiprobable (entre deux et sept) pour un routeur donné, le nombre *nbf* de feuilles de l'arbre est donné par :

$$nbf = \sum_{i=2}^{M-1}\left( N.\frac{a(i)}{6}\left( N.\sum_{j=2}^{M-1}\frac{a(j)}{6}\right)\right) = \frac{N^2}{36}\left(\sum_{i=2}^{7} a(i)\right)^2$$

où *a(i)* représente le nombre de feuilles de l'arbre élémentaire associé à un échange simple de ports d'un routeur possédant i voisins.

En étudiant un échange simple (entrée ou sortie) entre ports d'un routeur, on obtient la formule suivante :

$$a(i) = \sum_{k=0}^{i-1}\left(M-2-k\right)$$

En effectuant ce décompte, on élimine alors les feuilles correspondant au même échange, ainsi que les feuilles qui décrivent des échanges non réalistes entre ports non connectés. Pour les routeurs 8x8 ($M$=8, cas que l'on gardera lors de la plupart de nos tests par la suite), on obtient $nbf \approx 200\ N^2$. On conçoit qu'il est impossible de parcourir entièrement et naïvement un tel arbre. Il faut donc en examiner une partie judicieuse où les meilleures solutions ont de grandes chances de se trouver. Pour cela, nous allons tout d'abord réaliser des restrictions sur les ports intervertis (en entrée comme en sortie). Ainsi, nous allons garder en mémoire, lors des deux premières descentes de voisinage ($N_1$ et $N_2$) les cinq « moins mauvaises » permutations de chaque routeur (en entrée et en sortie). Nous entendons par « moins mauvaises » celles qui ne dégradent pas ou peu la solution courante. Aucune permutation ne peut améliorer seule la solution car on a

rencontré des échecs à toutes les tentatives sur les voisinages $N_1$ et $N_2$ avant de parcourir le voisinage $N_3$.

En effectuant la combinaison entre permutations « moins mauvaises », on a déjà réduit le nombre de feuilles de l'arbre de recherche à 25 $N^2$. Le routage effectué étant statique (réalisé avant l'arrivée du trafic réel), nous pouvons conserver ce voisinage, en sachant tout de même qu'il sera très coûteux en temps de calcul. Cependant, dans le cadre de nos expérimentations, nous avons voulu tester un grand nombre de réseaux pour chaque configuration. Nous avons donc voulu garder des temps de parcours comparables pour nos trois voisinages, afin que l'exécution du programme total ne dépasse pas deux heures de temps. Il a donc fallu à nouveau restreindre la recherche. Nous avons choisi de limiter le nombre de routeurs explorés dans l'arbre de recherche. Pour conserver une homogénéité entre les voisinages (en temps de parcours), nous avons restreint le voisinage à $\lfloor \sqrt{N} \rfloor$ routeurs (choisis aléatoirement parmi tous les routeurs), soit 7 dans le cas d'un réseau de 50 nœuds.

À ce stade, il y a des risques que l'on achève de visiter ce nouveau voisinage sans avoir amélioré notre solution. Si tel est le cas, on relance l'algorithme avec comme nouvelle solution la solution dernièrement obtenue. On introduit alors un compteur du nombre d'itérations effectuées, une itération correspondant à un échec sur tous les voisinages ou à une amélioration. Le critère d'arrêt est alors un nombre maximal d'itérations que nous fixons. Pour éviter de cycler sur une solution courante sans jamais améliorer le routage, on réalise si possible un mouvement qui laisse inchangée la fonction coût de la solution dans le cas où aucune amélioration n'a été trouvée sur le voisinage courant. Ainsi, la comparaison de la solution courante et de l'étude de son voisinage débouche sur trois possibilités :

- il existe une meilleure solution dans le voisinage et nous nous y rendons ;
- il n'existe que de pires solutions et nous testons un autre voisinage sur la même solution courante ;

- il existe au moins une solution équivalente (même nombre de connexions réalisées) et aucune meilleure solution, et nous nous déplaçons sur une des solutions équivalentes (choisie aléatoirement).

Ceci permet au programme de se déplacer sur des plateaux de solutions équivalentes en se donnant ainsi la possibilité de ne pas stagner trop longtemps près de minima locaux.

Finalement, le troisième voisinage sera parcouru durant un temps raisonnable, et à partir de solutions initiales différentes. Nous pensons ainsi parcourir de façon efficace le voisinage que nous avons introduit. La version définitive de notre algorithme, l'algorithme VNSFOR (Variable Neighbourhood Search For Optical Routing) est présenté à la Figure 3.5.

Pour explorer les voisinages ainsi définis, il faut être capable de sonder rapidement la qualité d'une solution courante. Nous avons pour cela mis en place un re-routage incrémentiel que nous présentons ci-après.

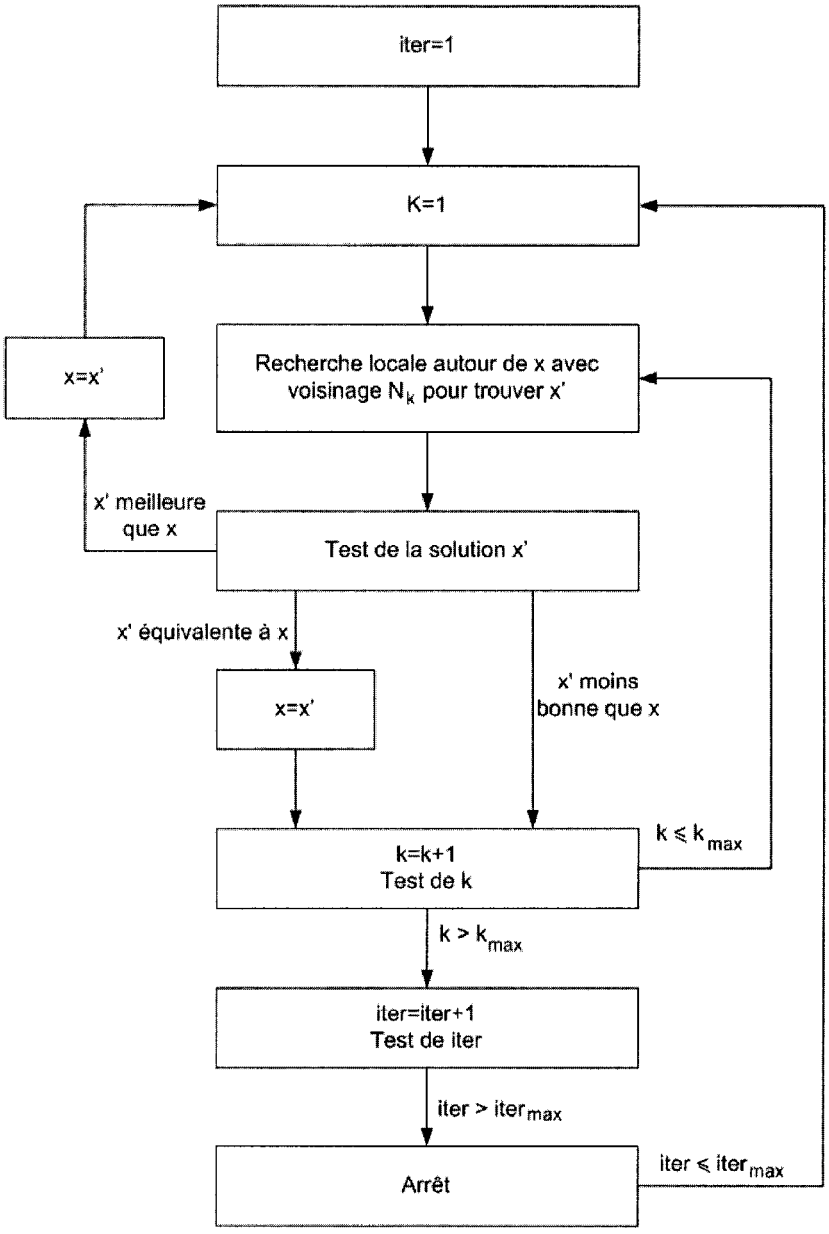



**Figure 3.5** Algorithme VNSFOR

### 3.3 Recherche à voisinage variable

Désirant avoir des voisinages les plus grands possibles, nous nous devons de présenter une méthode rapide d'évaluation de la qualité d'une feuille (évaluation du nombre de connexions établies avec la solution courante par exemple pour le premier

scénario). Lors de l'initialisation, le programme calcule la totalité des connexions engendrées par la solution initiale. Cette méthode est coûteuse car, pour chaque routeur, chaque port de sortie émetteur et chaque longueur d'onde, elle construit le chemin optique pour trouver la destination de la connexion. Il faut pouvoir exploiter ces données déjà calculées pour alléger l'étude de voisinage. En vue d'utiliser une méthode de recherche locale à voisinage variable, nous avons mis en œuvre une méthode incrémentielle qui se base sur les calculs de la solution courante déjà effectués pour évaluer la qualité de ses voisins.

**Méthode incrémentielle de re-routage**

En se focalisant sur les mouvements que l'on projetait de faire (échange de connexions en entrée, échange de connexions en sortie, échange multiple de connexions combinant plusieurs permutations en entrée et plusieurs permutations en sortie), nous avons dégagé une caractéristique commune à tous ces mouvements. Ils sont composés d'une (ou de plusieurs) suppression(s) de liens et (ou) d'un (ou de plusieurs) ajout(s) de liens. Par exemple, à la Figure 3.6, nous montrons comment une interversion de ports d'entrée correspond à la suppression de deux liens, et à la création de deux nouvelles liaisons (dans le cas d'un échange entre deux ports connectés).

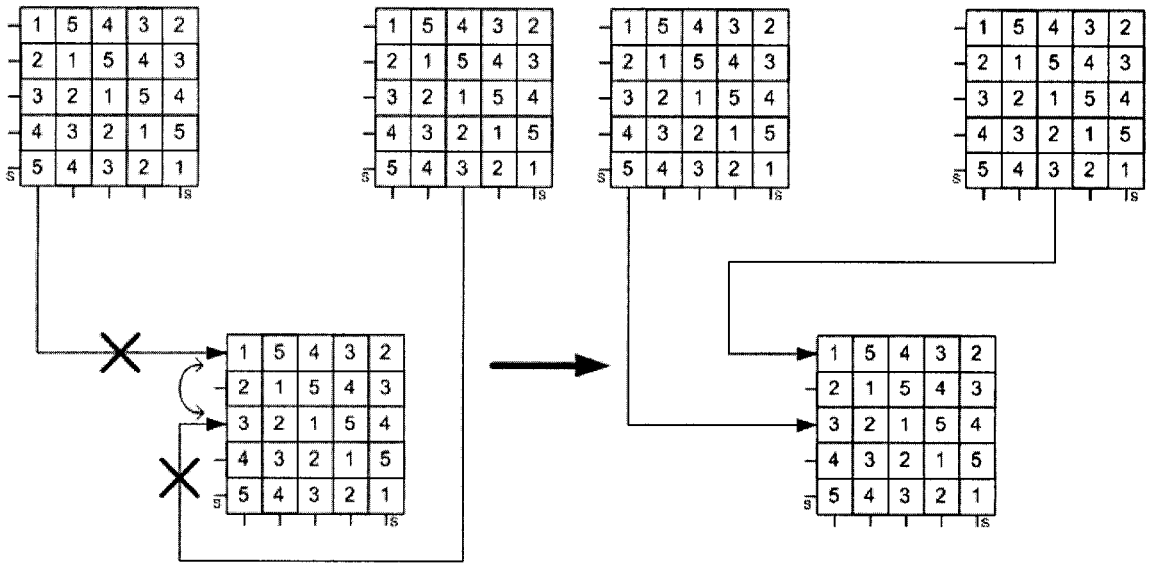


**Figure 3.6** Décomposition d'un mouvement en éléments simples

Il suffit donc d'implémenter le re-routage pour une suppression et pour un ajout de lien.

**Suppression de lien**

De manière assez surprenante de prime abord, le seul mouvement de suppression de liens a des conséquences assez importantes sur le routage dans le réseau optique.

Nous pouvons dégager trois grandes catégories de cas :

- destruction des chemins qui transitaient par le lien concerné (cas a);
- l'ancien port de sortie devient non connecté, donc il est maintenant capable de terminer un chemin optique : possible création de connexion (cas b);
- l'ancien port d'entrée devient lui aussi non connecté. En le reliant à un laser, il devient à son tour port émetteur et peut ainsi créer de nouvelles connexions (cas c).

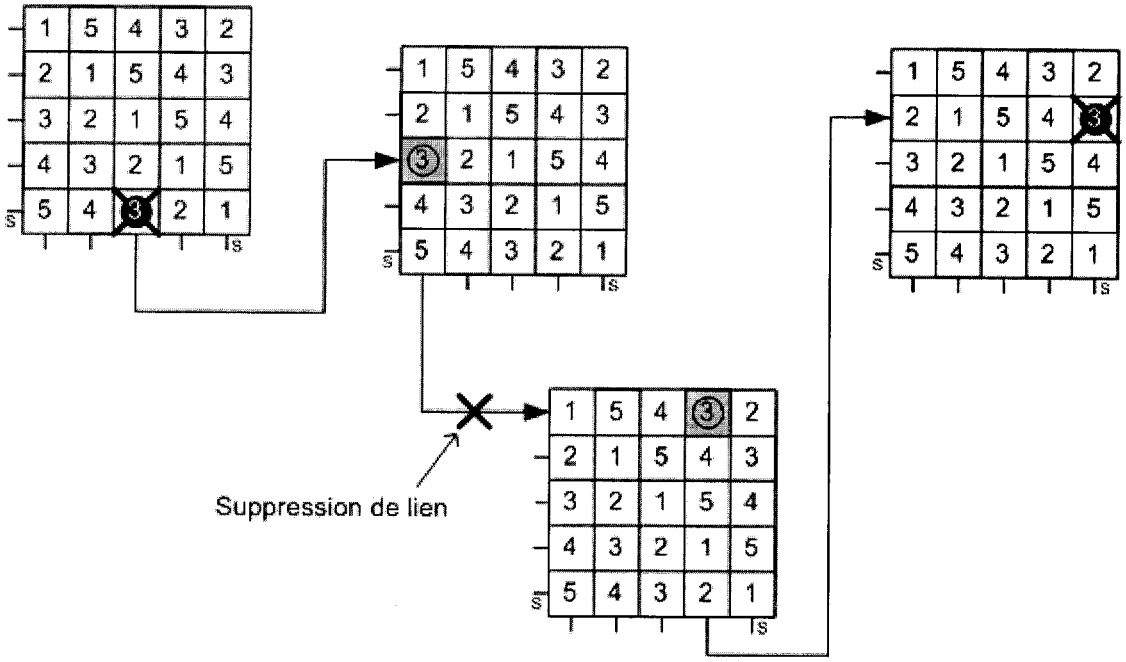



**Figure 3.7** Suppression d'un lien, cas a

Nous détaillons un exemple, aux Figures 3.7 et 3.8, où les trois effets introduits ici se réalisent. Par souci de clarté, les routeurs sont à nouveau représentés de petite taille mais il faut garder à l'esprit qu'ils sont en réalité de taille plus conséquente. Il faut bien noter ici que nous représentons les effets produits sur une seule longueur d'onde. Il faut procéder de même avec les autres longueurs d'onde. De plus, il existe des configurations où les cas b et c peuvent ne pas se produire. On ne retrouve pas le cas b lorsque le port d'entrée (correspondant à la longueur d'onde étudiée) du routeur en amont de la suppression du lien n'est pas connecté. Symétriquement, si le port de sortie (correspondant encore à la longueur d'onde étudiée) du routeur en aval de la suppression n'est pas connecté, le cas c n'a pas lieu.

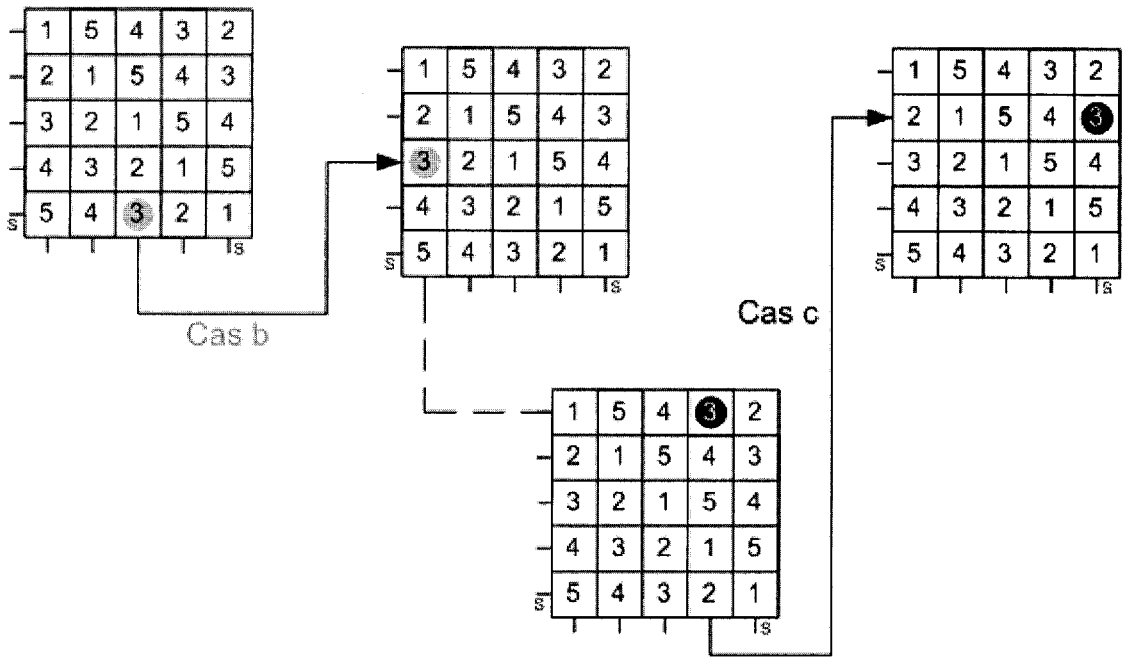



**Figure 3.8** Suppression d'un lien, cas b et c

Nous avons été confrontés, durant l'implémentation, à un cas de figure très particulier qui s'est trouvé être assez difficile à résoudre avec notre modélisation. Il se rencontre lorsqu'il y a formation de « boucles » à l'intérieur du réseau. Nous exposons à la Figure 3.9 un exemple très simple d'un tel phénomène; il peut s'avérer être plus complexe.

Nous remarquons que la longueur d'onde 3 du premier port de sortie du routeur supérieur ne pourra jamais être utilisée car elle se renvoie à elle-même après une boucle dans le réseau. Cela pose un problème si nous essayons d'ôter un des liens qui composent la boucle. Nous nous servons de la variable *routeur_origine* pour trouver le routeur source du chemin optique à modifier. Or, pour une telle situation, la variable *routeur_origine* n'est pas définie (il en est de même pour la variable routeur_destination). Nous avons donc fait en sorte que les variables *routeur_origine* (et routeur_destination) soient instanciées à nul dans un tel cas lors de la phase d'initialisation. Il faudra cependant être vigilant dans la phase d'ajout de lien quand on

crée une telle boucle. Il faudra que l'on soit capable de la détecter, et de fixer les variables *routeur_origine* (et routeur_destination) correspondantes à nul.

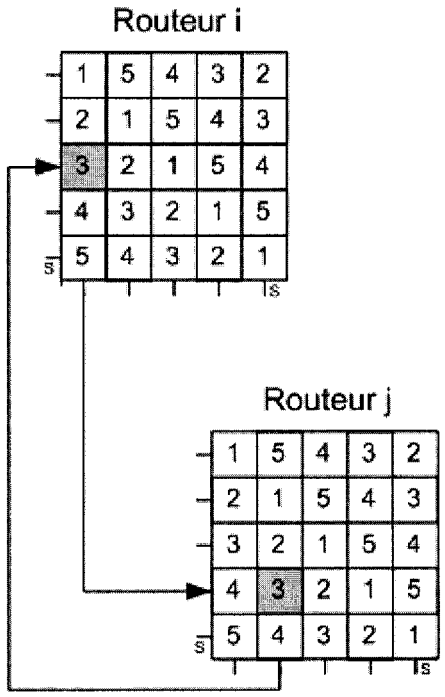



**Figure 3.9** « Boucle » dans un réseau

**Ajout de lien**

Le second élément de décomposition de n'importe quel mouvement complexe est donc l'ajout d'un lien. Son étude possède quelques similitudes avec l'élément précédent. Elle est à nouveau caractérisée par trois sortes de cas :

- -création d'une connexion entre un routeur en amont et un routeur en aval de la nouvelle liaison (cas a);
- -cependant, le nouveau port de sortie de la liaison devient connecté, donc n'a plus les capacités de réception d'un chemin optique (cas b);
- -de même, le nouveau port d'entrée n'est plus relié à un laser émetteur. Il n'est donc plus la source de certains chemins qu'il produisait (cas c).

Nous représentons l'ensemble de ces cas à la Figure 3.10.

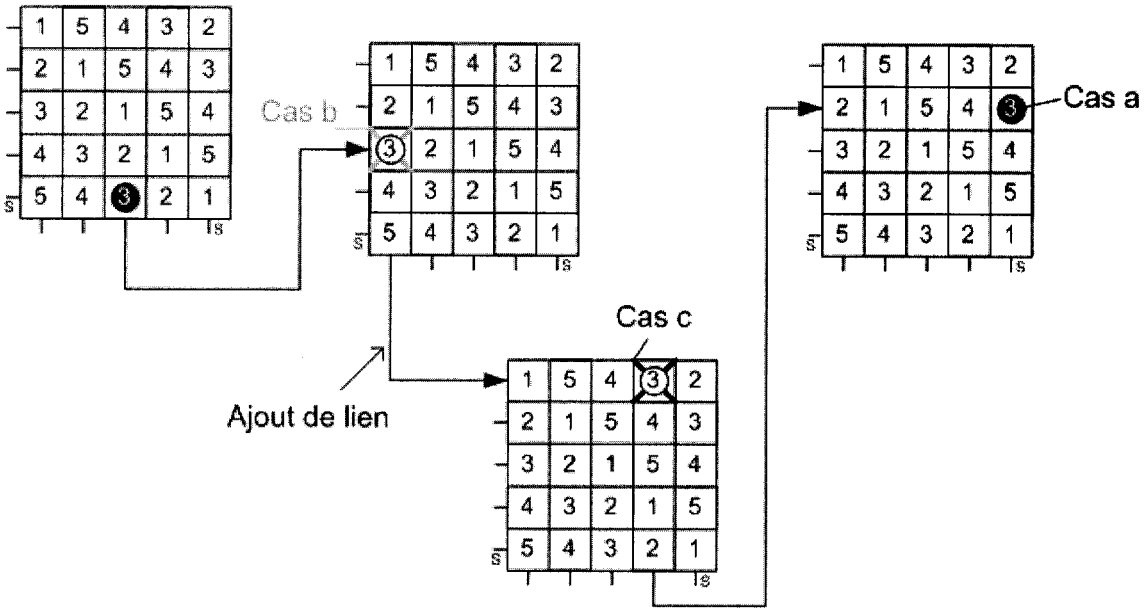



**Figure 3.10** Ajout d'un lien, cas a, b et c

Comme vu précédemment, il faut rester vigilant quant aux apparitions de boucles dans le réseau. Elles seront détectées en fait lors de la réalisation du c (et a en parallèle). Lorsque l'on va suivre le chemin optique pour trouver l'ancienne destination à modifier, nous allons contrôler que nous ne transitons jamais par le nouveau lien mis en place. Si tel était le cas, il faut s'assurer de mettre à nul les variables *Routeur_origine* correspondantes pour cette boucle. Nous détaillons cette possibilité à la Figure 3.11. Après l'ajout du lien, nous supprimons la connexion du routeur 3 vers le routeur 2 sur la longueur d'onde 3. Nous conservons en mémoire le fait que le chemin supprimé parte du routeur 3, port d'entrée 1. Nous devons alors poursuivre l'ancien chemin optique arrivant au routeur 2 par le port d'entrée 3 (longueur d'onde 3) qui avait pour *Routeur_origine* le routeur 3. En suivant le nouveau lien, nous arrivons sur le port d'entrée 1 du routeur 3. Nous comparons ceci avec l'information conservée en mémoire. Ayant détecté la similitude, l'algorithme repère la boucle et affecte la valeur nulle à toutes les variables suivantes: *Routeur_origine*(1,2,3) = 0, *Routeur_origine*(2,2,3) = 0, *Routeur_origine*(3,1,3) = 0 et *Routeur_origine*(4,1,3) = 0.

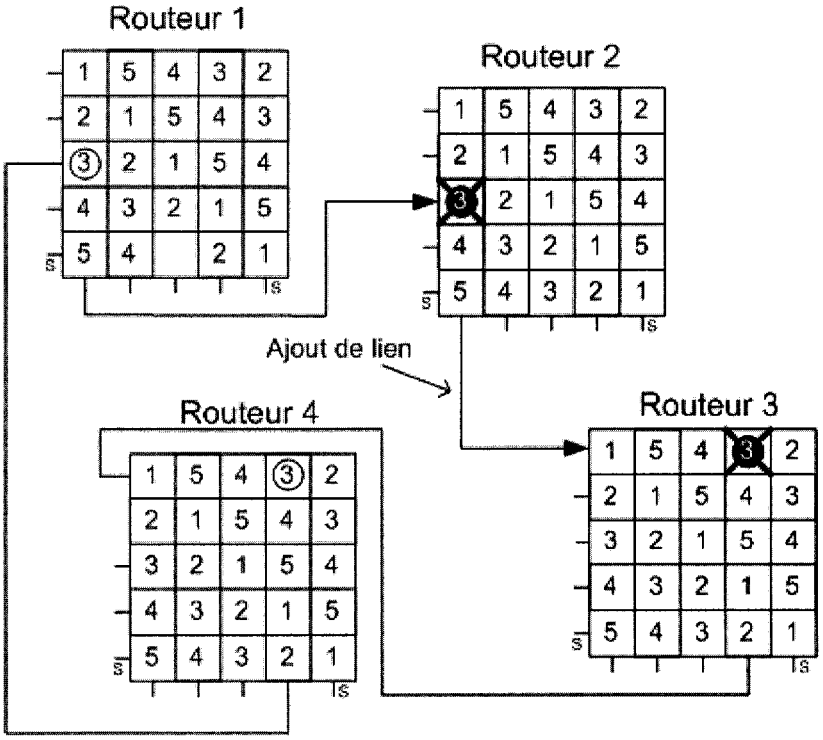



**Figure 3.11** Repérage de la création d'une boucle

Une fois cette analyse rapide et efficace du voisinage mise en place, nous pouvons définir la recherche à voisinage variable que nous proposons dans le cadre de ce mémoire.

### 3.4 Générateurs de réseaux

Une fois l'algorithme mis en place, il nous reste à construire un générateur de réseaux performant. La performance est jugée ici suivant plusieurs critères : il faut que ces réseaux soient réalistes, très variés, non spécifiques, pour proposer un jeu de données non truqué. Étant donné que nous avons voulu comparer les performances de notre algorithme avec celui proposé en [1], nous avons tout d'abord généré les mêmes réseaux tests que ceux qu'ils proposent dans leur article. Nous introduirons ensuite des réseaux qui correspondent certainement davantage aux réseaux classiques que l'on rencontre réellement.

### 3.4.1 Générateur classique de réseaux

Les réseaux utilisés pour les tests de l'algorithme LONCA présentent les caractéristiques suivantes :

- tous les nœuds du réseau sont au moins de degré deux ;
- le degré maximal d'un nœud est sept, car on utilise des routeurs 8x8. Avec le port virtuel qui doit nécessairement rester libre, il y a sept ports au maximum disponibles pour les connexions ;
- tous les degrés entre 2 et 7 sont équiprobables (donc ont la probabilité $\frac{1}{6}$). En moyenne, il existe autant de nœuds avec 2 voisins, que de nœuds avec 3 voisins, et ainsi de suite...

Le point négatif qui se dégage de suite de ces caractéristiques est le non contrôle de la connexité du réseau. En générant aléatoirement de tels réseaux, on pourrait très bien créer des réseaux non connexes, c'est-à-dire formés de plusieurs portions disjointes, et donc non connectables entre elles. Il est évident que de tels réseaux ne traduisent absolument pas les réseaux sur lesquels on souhaite appliquer les algorithmes de routage optique. Cependant, dans le souci de se baser sur les mêmes éléments de comparaison, nous avons généré de tels réseaux. Nous avons pour cela utilisé la programmation par contraintes, en utilisant la version 4.4 de *ILOG Solver*.

La recherche des réseaux se fait en deux temps. Tout d'abord, le programme affecte de manière équiprobable de deux à sept voisins à chacun des routeurs du réseau. Dans un second temps, il faut déterminer quels sont les voisins de tous les routeurs. Cela revient à positionner des liens bidirectionnels entre les nœuds, sous la contrainte de respecter le nombre de voisins de chacun. C'est la phase de création de la topologie du réseau. On introduit alors, comme variable du problème, une matrice carrée *connectes*[i][j], de taille *N*x*N*. L'élément [i][j] de cette matrice vaut 1 si *i* est connecté à *j* par un lien direct, 0 sinon. On impose tout d'abord que les éléments diagonaux de la matrice soient tous nuls car un routeur ne peut pas être connecté avec lui-même.

Cela s'écrit :

$$\forall(i \in [1...N]) \quad connexion[i][j] = 0$$

Les liens étant tous bidirectionnels, la matrice est tout d'abord symétrique. La deuxième contrainte s'énonce alors :

$$\forall(i \in [1...N], j \in [1...N], i < j) \quad connexion[i][j] = connexion[j][i]$$

La troisième contrainte impose le respect du nombre de voisins. Appelons *nbVoisins*[i] le vecteur qui renvoie le nombre de voisins calculé précédemment. On remarque que le produit scalaire de la ième matrice ligne de *connectes*[i][j] avec le vecteur unitaire composé exclusivement de 1 et de taille $N$ est égal au nombre de voisins. On pose donc :

$$\forall(i \in [1...N] \quad connexion[i].[11...1] = nbVoisins(i)$$

L'algorithme affecte alors aléatoirement les éléments *connectes*[i][j] avec des 1 ou des 0. A chaque affectation, la propagation sur les contraintes proposées réduit les domaines de certaines variables *connectes*[i][j] en les forçant à la valeur 1 ou 0. Nous avons cependant remarqué que les affectations peuvent mener à des impasses, et qu'il faudrait un très grand nombre de retours arrière (backtrack) pour parvenir à une solution réalisable. Nous avons préféré imposer un nombre maximal de retours arrière au-delà duquel l'algorithme d'affectation de *connectes*[i][j] se relance. Cette modification s'est trouvée très utile pour les grands réseaux ($N$>80).

### 3.4.2 Générateur évolué de réseaux

L'aspect encore perfectible du générateur de réseaux précédent est la trop grande liberté laissée aux réseaux créés. Il est évident que les réseaux réels sont soumis à énormément plus de contraintes. Le choix de la programmation par contraintes s'est donc avéré très intéressant lors de cette étape. En effet, la grande souplesse d'utilisation de la programmation par contraintes permet de rajouter de nouvelles contraintes, indépendamment les unes des autres, sans avoir à repenser le corps du programme. Notre première volonté a été de garantir la connexité du réseau. Nous avons pensé introduire une contrainte sur les puissances de la matrice *connectes*[i][j]. En effet, si on

prend le carré de la matrice *connectes*[i][j], on obtient des éléments non nuls en [i][j] s'il existe au moins un chemin de longueur inférieure ou égal à 2 qui relie $i$ à $j$. Par récurrence, l'élément [i][j] de la puissance kième de la matrice *connectes* est non nul si et seulement s'il existe au moins un chemin reliant $i$ à $j$ de longueur $2^k$. En prenant $k$=4 pour des réseaux de 50 nœuds, nous pouvions assurer la connexité du réseau. Cependant, cette contrainte nous est apparue plus mathématique que réaliste. Dans les réseaux existants, on a bien plus qu'une simple connexité entre les nœuds. En particulier, les réseaux optiques sont généralement conçus à partir de squelette en forme d'anneau sur lequel on développe le réseau.

Pour représenter ceci, nous avons simplement choisi d'imposer à tous les réseaux de présenter une ossature commune : une liaison circulaire qui lie directement le routeur 1 au routeur 2, le routeur 2 au routeur 3, et ainsi de suite jusqu'au routeur $N$ avec le routeur 1. Ceci s'implémente de manière très facile à partir du modèle proposé plus haut. Il suffit de partir d'une matrice *connectes* déjà partiellement affectée. La description de l'anneau correspond exactement, d'après notre modélisation, à la matrice compagnon (matrice notée $H$) bien connue des mathématiciens, aussi connue sous le nom de matrice des permutations. C'est une matrice carrée composée de 1 au dessus de la diagonale, ainsi que dans le coin gauche inférieur, et de 0 partout ailleurs, comme exposé dans un petit cas ($N$=4) à la Figure 3.12.

$$H = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

**Figure 3.12** Matrice compagnon

Un élément qui apparaît encore insatisfaisant dans notre génération de réseaux et qui mériterait une étude plus approfondie réside dans le caractère plan des réseaux réels.

En effet, étant disposés à la surface terrestre, les routeurs sont confinés dans un plan de l'espace. Cependant, notre algorithme produit des réseaux qui s'affranchissent totalement de la notion de distance induite par la disposition plane du réseau. Notre algorithme ne respecte absolument pas la règle tacite : « les amis de mes amis sont souvent mes amis » ou plus simplement « les voisins de mes voisins sont souvent mes voisins ». Vouloir respecter cette particularité serait sans doute un élargissement très intéressant de notre travail : générer des réseaux de plus en plus réalistes en exhibant des contraintes adéquates pour notre modèle s'avérerait être un complément fructueux de notre recherche.

# CHAPITRE IV
# PROTOCOLE EXPÉRIMENTAL ET RÉSULTATS

Ce chapitre présente les résultats que l'on a obtenus en utilisant notre algorithme. Il expose en premier lieu le protocole expérimental précis que l'on s'est proposé de faire avant de se lancer dans les simulations. Nous justifierons ainsi les choix que nous avons opérés dans nos tests et nous mettrons en avant leur pertinence. Nous y rapporterons les différentes difficultés qui sont apparues au cours de la programmation et les méthodes que l'on a développées pour y faire face. Nous traiterons seulement dans ce chapitre de l'algorithme de recherche locale à voisinage variable, l'autre méthode utilisant la programmation par contraintes n'ayant pas porté ses fruits. Toutes les implémentations ont été réalisées au moyen de *ILOG Solver*, librairie de C++ particulièrement adaptée à la programmation par contraintes. En effet, nous avions l'intention au préalable d'utiliser exclusivement la programmation par contraintes dans le cadre de notre recherche et *ILOG Solver* semblait le choix logique. Par la suite, nous avons simplement conservé des fonctionnalités d'*ILOG Solver* pour la génération de réseaux aléatoires et le parcours des arbres de recherche, mais le reste de la programmation est réalisée en C++.

Dans un second temps, nous détaillerons les résultats de l'algorithme final (VNSFOR pour Variable Neighbourhood Search For Optical Routing) suivant l'ordre du protocole expérimental. Nous les confronterons à ceux de l'algorithme LONCA. Nous mettrons finalement en avant des réseaux plus réalistes pour dresser des performances plus significatives.

## 4.1 Protocole expérimental

Avant de réaliser toute série de simulations visant à évaluer la méthode proposée, il faut définir précisément un protocole de tests qui mettront en valeur les critères pertinents que nous déciderons de considérer. Ainsi, nous devons mettre en valeur les

données du problème auxquelles nous avons accès. Nous présenterons alors celles que nous analyserons précisément et dans quelle mesure nous les ferons varier.

### 4.1.1 Paramètres du problème

Tel que nous avons défini la problématique au chapitre 3, nous étudions des réseaux optiques uniquement composés de routeurs latins, dont la topologie est fixée. Nous établissons ici une liste exhaustive de tous les paramètres qui rentrent en jeu dans notre étude :

- fonction objectif à optimiser;
- caractéristiques du réseau;
- caractéristiques des voisinages;
- caractéristiques de l'algorithme.

Pour chacun d'eux, nous allons proposer les tests pertinents qui nous assureront une étude précise du problème général.

### 4.1.2 Fonction objectif

Nous avons présenté dans le chapitre 3 les deux fonctions objectifs que nous souhaitons optimiser suivant les deux scénarii envisagés. La différence entre les deux réside dans le choix du voisin de la recherche locale. Étant donné qu'il est facile de ne modifier que cette « fonction coût » évaluant la qualité d'une solution courante, nous avons tout d'abord choisi de nous concentrer seulement sur le premier scénario envisagé (maximiser le nombre total de connexions possibles). Nous ferons ainsi nos différents tests en suivant cette logique, et nous présenterons à la fin les résultats obtenus sur le second scénario.

### 4.1.3 Caractéristiques du réseau

Nous distinguons ici les caractéristiques du réseau et celles des routeurs.

Pour tenter de reproduire le plus fidèlement possible les cas réels, il faut choisir une taille de réseaux sensée. Les liaisons optiques étant souvent utilisées dans les dorsales de grands réseaux, nous avons souhaité prendre des valeurs comparables à de tels exemples. Ainsi, si l'on se réfère au réseau NSFNET (et ses 17 nœuds) qui couvrait l'Amérique du Nord jusqu'en 1995, notre algorithme doit pouvoir optimiser des réseaux comprenant plusieurs dizaines de nœuds. Nous avons donc décidé de tester des réseaux ayant entre 50 et 100 nœuds. Une fois le nombre de nœuds fixé, il faut encore s'attarder sur leur répartition. Dans l'optique de comparer nos résultats avec ceux de l'algorithme LONCA, nous avons utilisé les mêmes critères pour établir le degré moyen des nœuds du graphe (nombre moyen de voisins d'un nœud). Ainsi, les différentes valeurs possibles pour le degré d'un nœud sont équiprobables (sauf la valeur 1 qui est ôtée car elle ne correspond pas à la réalité). Ainsi, si l'on considère des routeurs à huit ports (routeurs latins 8x8), les valeurs possibles pour le degré d'un nœud sont comprises entre deux et sept (il existe au moins un port non connecté pour permettre l'émission et la réception de chemins optiques). Un exemple de vecteur aléatoire représentant les différents degrés de tous les nœuds d'un réseau de 50 nœuds est proposé à la Figure 4.1.

```
Le vecteur des degrés est : [4 6 7 5 4 7 2 3 5 7 3 2 3 4 7 6 6 5 5 5 6 4 6 6 2 7 6 2 2 2 2 2 4
                             4 2 6 7 3 4 5 4 3 2 5 5 7 6 5 6 5]
```

**Figure 4.1** Vecteur des degrés d'un réseau de 50 nœuds

Dans un second temps, il faut préciser les routeurs que l'on utilise. Dans toutes nos études, nous prenons pour référence des routeurs 8x8 (donc, avec $M$=8). Tous les routeurs des réseaux sont pris identiques. Les tests seront toujours effectués sur un panel de 100 réseaux générés ainsi.

### 4.1.4 Caractéristiques des voisinages

La définition des voisinages a été effectuée dans le chapitre précédent. Nous détaillons ici les caractéristiques des voisinages simples, puis celles du voisinage plus complexe.

**Programmation des voisinages simples**

Lors de cette première phase de programmation, nous nous sommes limités à la maximisation du nombre de connexions réalisées dans le réseau optique. Nous avons tout d'abord implanté les deux premiers voisinages de la recherche à voisinage variable. Ceux-ci correspondent aux mouvements simples de permutation de ports de sortie (premier voisinage) ou d'entrée (deuxième voisinage) présentés dans le chapitre précédent. Nous avons pour ceci utilisé comme variable de décision le vecteur *vars* de dimension trois correspondant à la profondeur de l'arbre de recherche généré par ces mouvements locaux. Le principe général de la méthode consiste à affecter une valeur au vecteur *vars* (correspondant à une permutation réalisée) et à la tester. Si le mouvement améliore le nombre de connexions du réseau, ce mouvement est effectué : c'est le principe de « première amélioration ». Dans le cas contraire, on effectue un retour-arrière (backtrack) au niveau de l'arbre de recherche pour tester de nouvelles feuilles de l'arbre. Nous schématisons cette méthode à la Figure 4.2.

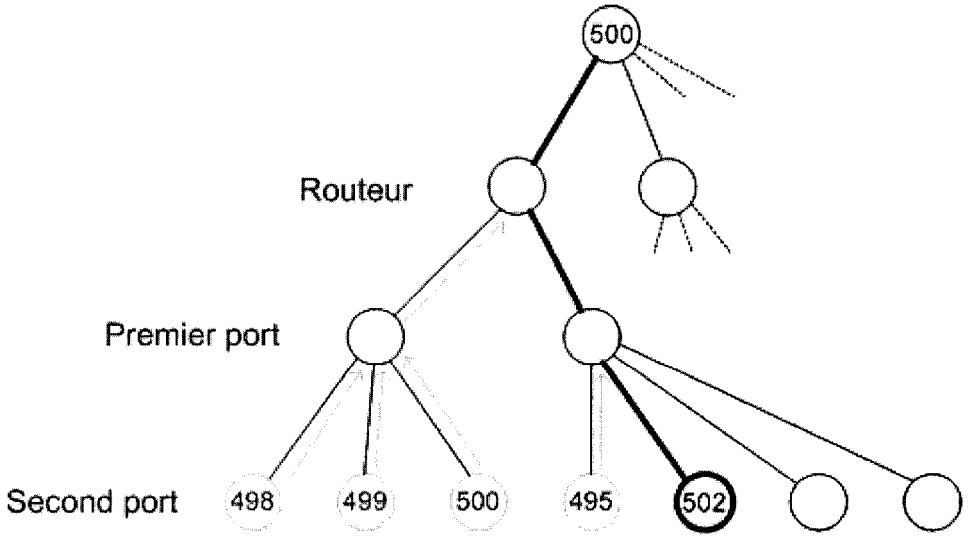



**Figure 4.2** Arbre de recherche des deux premiers voisinages

Le nombre affecté au nœud source représente le nombre de connexions établies par la solution courante. Les nombres positionnés à l'intérieur des feuilles représentent le nombre de connexions réalisées par chacun des voisins correspondants. Cette méthode est implémentée par le code de la Figure 4.3.

```
m.solve(IlcAnd(choixdunefeuille,
                testdelafeuille),
        IlcTrue);
```

**Figure 4.3** Programmation de l'arbre de recherche

La méthode *solve* prend en argument un (ou plusieurs) *IlcGoal*. Un *Goal* en *ILOG Solver* est une pierre fondatrice du grand édifice de la recherche de solution. Ici, *choixdunefeuille* affecte le vecteur *vars*. L'affectation se fait aléatoirement pour les routeurs et dans l'ordre lexicographique pour les ports (du port 1 au port $M$-1). Ensuite, le *Goal testdelafeuille* est appelé. Si le point du voisinage n'est pas satisfaisant, cela déclenche un *IlcGoalFail* qui a pour but de revenir à la dernière affectation de *vars* réalisée, d'enlever la valeur précédemment choisie de son domaine de définition et d'essayer une nouvelle valeur possible. Il faut concevoir ici que *vars* est un vecteur à trois dimensions : un retour en arrière correspond donc à la dernière affectation d'une composante de *vars* effectuée. Si la nouvelle solution améliore la fonction objectif, le solveur s'arrête avec les valeurs de ce voisin sauvegardées dans une variable auxiliaire. Le *IlcTrue* en deuxième argument du *solve* permet d'annuler les décisions prises à l'intérieur du *m.solve* sur *vars* une fois qu'on en sort. Ainsi, au prochain appel de *solve*, *vars* aura à nouveau la possibilité de parcourir entièrement l'arbre de recherche.

Pour juger de la qualité du premier voisin, la fonction *routage* est tout d'abord appelée pour instancier les variables du problème présentées dans le chapitre 3 :

- *Routeur_origine*[i][j][k] dénote le routeur à l'origine du chemin optique arrivant au routeur $i$, par le port d'entrée $j$ à la longueur d'onde $k$;

- *cx_routeur*[i][j] (et *cx_port*[i][j]) qui fournit le routeur (et le port correspondant) auquel est relié le port de sortie *j* du routeur *i*;
- *portsConnectesentree*[i][j] (et *portsConnectessortie*[i][j]) qui contient le *j*ème port d'entrée (et de sortie) connecté pour le routeur *i*;
- *Connexion*[i][j] dénombre les connexions de *i* vers *j*.

Par la suite, pour évaluer la qualité de ces voisins, l'algorithme utilise la méthode de re-routage incrémentiel.

**Taille des voisinages**

Les deux premiers voisinages étant de taille fixe, nous pouvons faire varier la taille du troisième. Comme proposé dans le chapitre précédent, nous avons choisi aléatoirement $\lfloor\sqrt{N}\rfloor$ routeurs dont les connexions sont permutées en entrée ainsi que $\lfloor\sqrt{N}\rfloor$ routeurs dont les connexions sont permutées en sortie.

### 4.1.5 Caractéristiques de l'algorithme VNSFOR

Le dernier facteur sur lequel on peut jouer au niveau de l'algorithme est le nombre d'itérations qu'on se propose de réaliser avant de sortir de la recherche locale. Il faut bien comprendre ici que notre logique d'implémentation diffère quelque peu de celle du programmeur qui aurait à optimiser un réseau réel. En effet, notre routage est un routage statique (non dynamique, donc réalisé avant la mise en place physique du réseau). Le facteur temps est donc très peu limitatif et l'ingénieur pourrait allouer à son algorithme un temps relativement conséquent. Cependant, durant notre étude, nous sommes contraints de produire un algorithme plus court (des temps de calcul variant généralement entre 1h et 2h) car nous réalisons des résultats sur un grand nombre de réseaux. Le facteur temps apparaît alors comme limitant, et va guider le choix du nombre d'itérations. Nous avons envisagé deux cas suite aux premiers résultats obtenus par notre algorithme. Le premier (correspondant par la suite à l'algorithme VNSFOR)

utilise un nombre d'itérations constant égal à 150. Le second (correspondant par la suite à l'algorithme VNSFOR2) utilise un nombre d'itérations fonction linéaire de la taille du réseau routé. Ce nombre d'itérations est posé égal à trente fois le nombre de nœuds du réseau. Ce facteur a été choisi de façon empirique et pourrait constituer une voie d'approfondissement de notre recherche.

### 4.1.6 Protocole expérimental

Une fois dégagés les paramètres significatifs que l'on compte faire varier durant nos tests, nous pouvons mettre en avant, à la Figure 4.8, l'organisation précise des expériences que l'on compte mener.

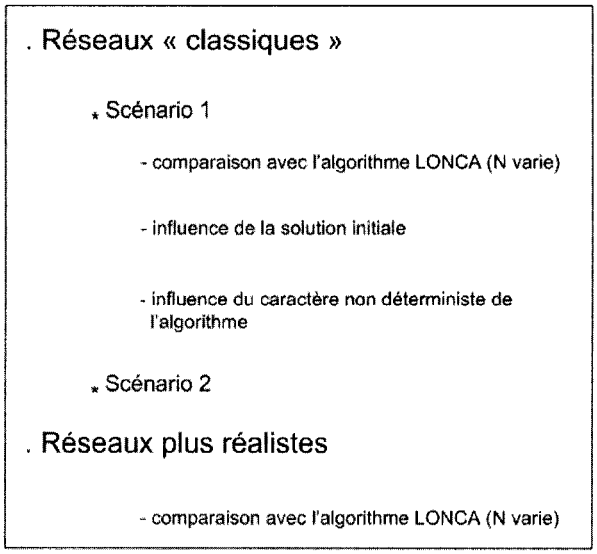



**Figure 4.8** Protocole expérimental

## 4.2 Tests sur les réseaux « classiques »

En suivant le protocole expérimental précédemment établi, nous nous intéressons en premier lieu à nous comparer à un algorithme référence avec les paramètres conformément utilisés. Nous allons donc analyser les deux scénarii sur des réseaux tels qu'ils sont habituellement générés dans la littérature.

### 4.2.1 Premier scénario : comparaison avec l'algorithme LONCA

Nous nous intéressons tout d'abord à la résolution du même problème que précédemment, à savoir maximiser le nombre total de connexions dans le réseau, pour des réseaux optiques « classiques » (tels que définis dans [1]). Les premiers résultats comparatifs entre notre algorithme et l'algorithme LONCA sur un réseau de 50 nœuds sont exposés au Tableau 4.1. Notre algorithme VNSFOR opère ici sur un nombre constant de 150 itérations.

**Tableau 4.1** Tableau comparatif des différents algorithmes

| | Moyenne (connexions) | Ecart type (connexions) | Médiane (connexions) | min (connexions) | max (connexions) | Nombre de réseaux testés |
|---|---|---|---|---|---|---|
| VNSFOR | 595,03 | 20,04 | 596 | 553 | 644 | 100 |
| LONCA | 564,76 | 17,17 | 565 | 511 | 616 | 500 |

Nous pouvons tout d'abord voir que les résultats sont nettement meilleurs en moyenne (plus de 5% d'amélioration par rapport l'algorithme LONCA), avec un écart type voisin légèrement supérieur. Nos résultats s'étalent autour d'une valeur moyenne proche de 600. Nous pouvons donc poursuivre ces tests en faisant varier le nombre N de routeurs du réseau. Ils sont représentés à la Figure 4.4. Les améliorations apportées par notre algorithme VNSFOR sont à la Figure 4.5.

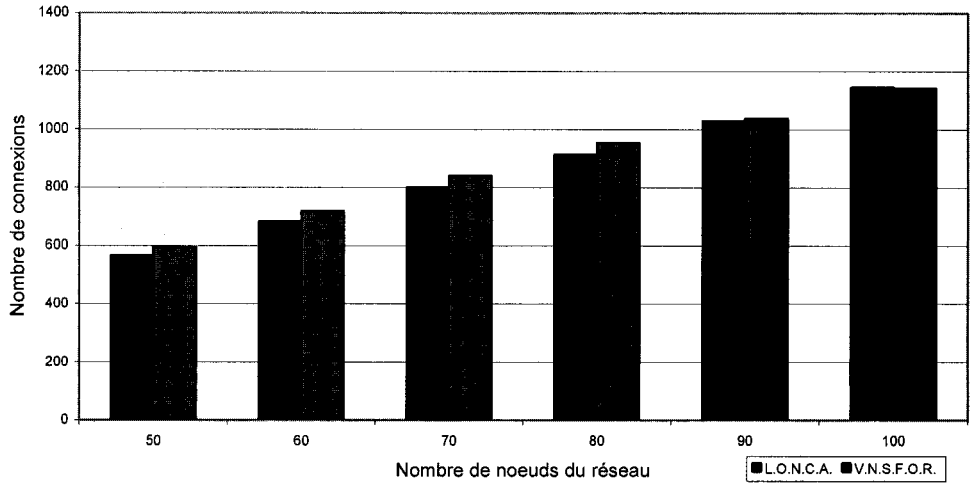



**Figure 4.4** Résultats comparatifs des algorithmes LONCA et VNSFOR

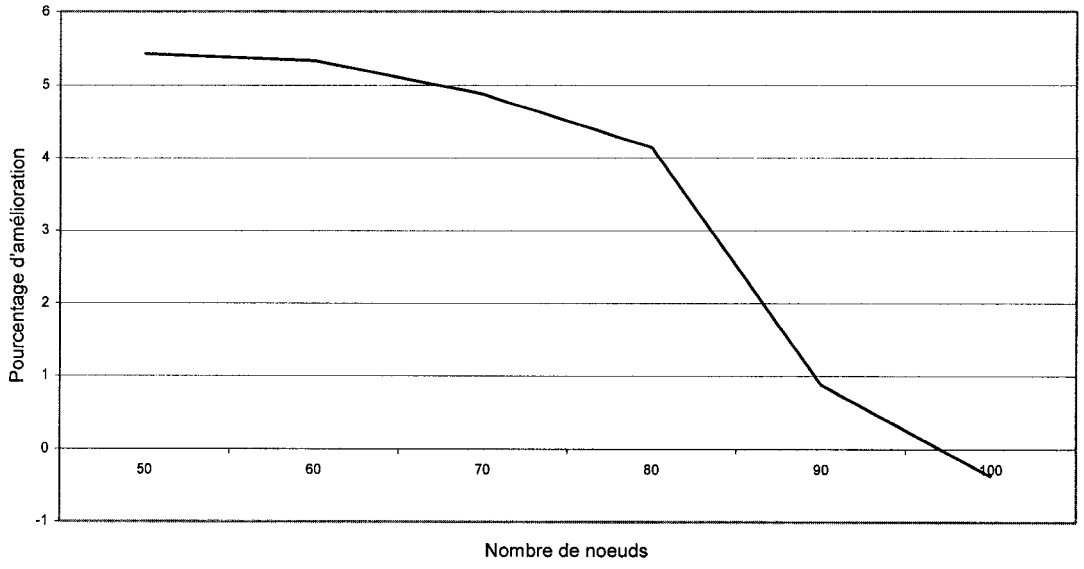



**Figure 4.5** Amélioration de VNSFOR

Les résultats sont répertoriés dans le Tableau 4.2 (l'algorithme VNSFOR2 sera présenté ci-après).

**Tableau 4.2** Comparatif détaillé des trois algorithmes

| Nombre de nœuds | Algorithmes | Moyenne (connexions) | Ecart type (connexions) | Min (connexions) | Max (connexions) | Médiane (connexions) |
|---|---|---|---|---|---|---|
| N=50 | LONCA | 564,76 | 17,17 | 511 | 616 | 565 |
| | VNSFOR | 595,43 | 20,57 | 553 | 644 | 596 |
| | VNSFOR2 | 595,43 | 20,57 | 553 | 644 | 596 |
| N=60 | LONCA | 681,87 | 18,14 | 631 | 742 | 683 |
| | VNSFOR | 718,27 | 20,69 | 663 | 766 | 720 |
| | VNSFOR2 | 723,16 | 21,69 | 663 | 774 | 721 |
| N=70 | LONCA | 799,7 | 20,32 | 744 | 863 | 799 |
| | VNSFOR | 838,70 | 20,86 | 782 | 898 | 837 |
| | VNSFOR2 | 848,70 | 21,77 | 799 | 902 | 849 |
| N=80 | LONCA | 914,24 | 21,51 | 855 | 982 | 915 |
| | VNSFOR | 952,06 | 20,29 | 899 | 999 | 953 |
| | VNSFOR2 | 965,83 | 21,79 | 892 | 1002 | 968 |
| N=90 | LONCA | 1027,33 | 21,91 | 968 | 1087 | 1035 |
| | VNSFOR | 1036,33 | 20,15 | 992 | 1070 | 1039 |
| | VNSFOR2 | 1078,65 | 22,60 | 1035 | 1117 | 1082 |
| N=100 | LONCA | 1144,54 | 24,82 | 1067 | 1230 | 1146 |
| | VNSFOR | 1140,33 | 20,63 | 1096 | 1188 | 1141 |
| | VNSFOR2 | 1189,97 | 21,26 | 1142 | 1232 | 1194 |

Deux réactions sont soulevées par ces différents schémas. On remarque tout d'abord que l'algorithme VNSFOR est très performant pour les réseaux de petite taille (plus de 5% d'amélioration en moyenne). Cependant, cette tendance se dégrade nettement pour les réseaux de taille plus importante. C'est alors que nous nous sommes interrogés sur la pertinence du choix d'un nombre d'itérations (pour les différents voisinages) constant. En effet, représentons la courbe du temps de calcul de notre algorithme en fonction du nombre de nœuds du réseau (Figure 4.6). Nous avons tracé ici le temps au bout duquel le critère d'arrêt est appelé, c'est-à-dire au bout duquel le nombre d'itérations atteint le nombre maximal autorisé.

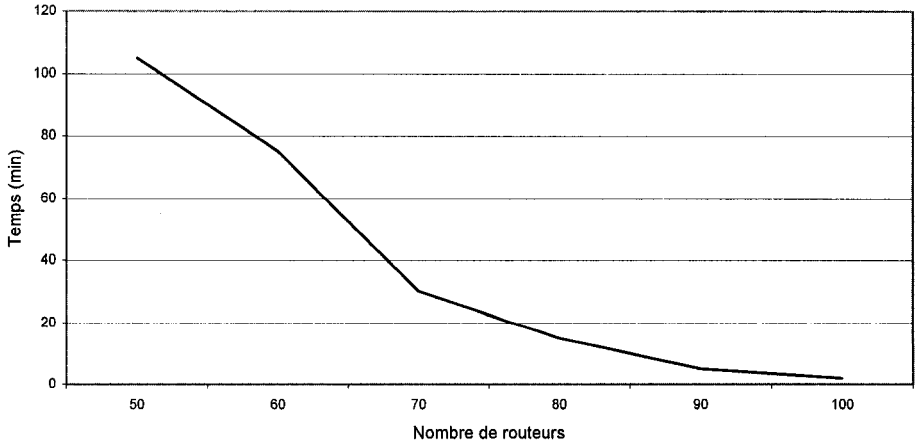



**Figure 4.6** Temps de calcul de VNSFOR, nombre d'itérations constant

L'algorithme est réalisé à nombre d'itérations constant. Une recherche rapide correspond alors à une optimisation qui a souvent détecté un bon voisin dès les premiers voisinages. Nous en avons déduit que l'algorithme VNSFOR passe trop peu de temps à explorer le troisième voisinage dans le cas de grands réseaux comparativement au cas de réseaux de petite taille. A notre avis, dans les réseaux de plus de 70 routeurs, l'algorithme trouve souvent des mouvements satisfaisants dès les deux premiers voisinages étant encore loin de l'optimum au bout de quelques dizaines d'itérations. Il faut donc permettre à notre algorithme d'exploiter la puissance de recherche du troisième voisinage en augmentant le nombre d'itérations permises relativement à la taille du réseau. Ainsi, le nombre d'itérations est fonction de la taille du réseau suivant la correspondance du Tableau 4.3 pour une évolution de notre algorithme appelée VNSFOR2.

**Tableau 4.3** Nombre d'itérations en fonction de la taille du réseau

| Nombre de noeuds | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|
| Nombre d'itérations | 150 | 180 | 210 | 240 | 270 | 300 |

Nous pouvons alors présenter successivement les nouveaux résultats obtenus aux Figures 4.7 et 4.8, ainsi qu'au Tableau 4.2 (ligne VNSFOR2).

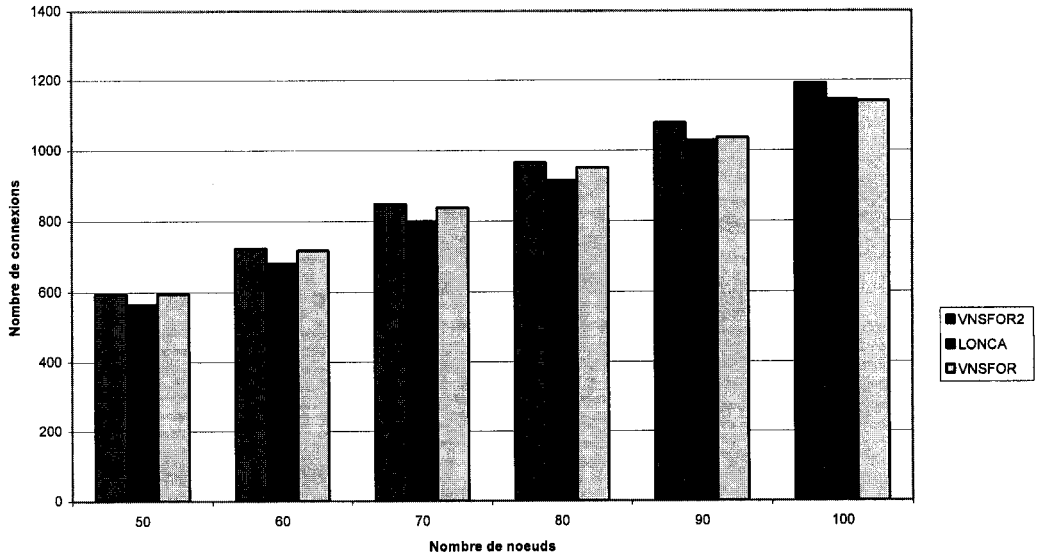



**Figure 4.7** Résultats comparatifs des trois algorithmes

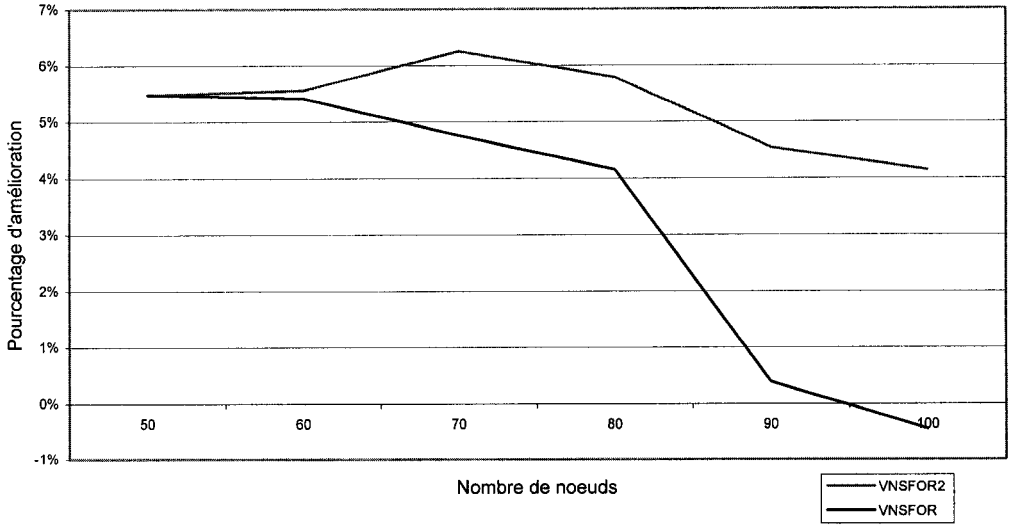



**Figure 4.8**. Améliorations comparées par rapport à l'algorithme LONCA

Nous pouvons donc constater que la deuxième version de notre algorithme de recherche locale à voisinage variable offre de meilleurs résultats que l'algorithme LONCA, et ce, quelle que soit la taille du réseau. En moyenne, nos résultats sont toujours meilleurs, de 5% environ. Les écarts type sont cependant comparables voire légèrement supérieurs à LONCA. Nous avons donc mis en œuvre un algorithme performant pour optimiser les connexions optiques dans un réseau composé de routeurs latins. Il faut cependant tester la robustesse de notre algorithme. En effet, en tant qu'heuristique, notre algorithme ne nous garantit jamais la qualité d'un résultat. Nous allons ainsi tenter d'évaluer, au moyen de tests judicieux, la confiance que l'on peut accorder aux résultats fournis par l'algorithme.

Deux critères déterminent la fiabilité de l'algorithme : l'influence du choix des voisins tout au long des itérations et l'influence de la solution initiale.

Pour évaluer l'impact des choix aléatoires effectués au cours de la recherche locale (ordre de parcours des routeurs, choix des routeurs retenus dans le troisième voisinage), nous avons lancé plusieurs fois l'algorithme sur le même réseau à partir de la même solution initiale. Nous consignons les résultats dans la première ligne du Tableau 4.4.

**Tableau 4.4** Robustesse de l'algorithme

| 100 cas avec | Moyenne (connexions) | Ecart type (connexions) | Min (connexions) | Max (connexions) | Médiane (connexions) |
|---|---|---|---|---|---|
| même solution initiale | 591,03 | 2,03 | 586 | 596 | 591 |
| solutions initiales aléatoires | 591,19 | 2,27 | 585 | 597 | 591 |

Nous réitérons l'opération à partir du même réseau et des solutions initiales aléatoires. Nous en déduisons tout d'abord que le choix d'une solution initiale particulière n'a quasiment pas de répercussions sur le résultat obtenu à l'issue des itérations. En effet, l'écart type est seulement augmenté de 0.24 connexions en choisissant une solution différente. Nous pouvons donc considérer que nos résultats sont

très peu dépendants de la solution initiale choisie. En d'autres termes, le choix d'une solution initiale ne nous confine pas dans une certaine partie de l'espace des solutions possibles. Ce critère est un point extrêmement intéressant lors de l'évaluation d'une heuristique telle que celle que l'on propose. Nous confrontons maintenant notre algorithme à l'algorithme LONCA dans le cadre du deuxième scénario envisagé : répondre le mieux possible à une matrice de demande de trafic.

### 4.2.2 Second scénario

Pour tester cette situation, nous avons tout d'abord implémenté un code qui génère aléatoirement des matrices de demande de trafic. Ces matrices sont caractérisées par le nombre $i$ de connexions désirées. Nous plaçons pour cela aléatoirement $i$ nombres un dans une matrice $N$x$N$ initialement composée uniquement de zéros. Tous nos tests seront ici effectués sur un réseau composé de $N$=50 nœuds. Il faut cependant demeurer vigilant avec une telle génération : nous pouvons créer des demandes qui sont impossibles à réaliser par nature, et ce, quel que soit le routage effectué dans le réseau. En effet, nous avons vu par exemple que la borne supérieure du nombre de connexions d'un routeur est 15 (dans le cas des routeurs latins de taille 8). Si la demande est donc supérieure à 16 connexions, elle ne pourra pas être réalisée de façon intrinsèque.

Pour étudier ce deuxième scénario, nous avons aussi dû adapter l'algorithme VNSFOR2. Pour ce faire, comme expliqué précédemment dans le chapitre 3, nous avons modifié la fonction d'optimisation en :

$$Max\left(\sum_{(i,j)\in[1,N]^2}\left(Demande(i,j)*connexion(i,j)\right)\right)$$

Nous comptons dorénavant seulement les connexions réalisées qui correspondent à la demande du concepteur de réseau. Ainsi, à chaque test de solution courante, c'est cette nouvelle fonction qui détermine la qualité de la solution obtenue.

Le Tableau 4.5 récapitule les différents résultats obtenus, comparativement à ceux présentés avec l'algorithme LONCA. Les pourcentages respectifs sont illustrés à la

Figure 4.9. Nous pouvons de suite remarquer que notre algorithme surpasse largement l'algorithme LONCA suivant ce scénario. Les contraintes sont ici plus restrictives : il est nécessaire de créer des connexions parmi un certain nombre de connexions désirées. Une recherche au hasard utilisant de petits voisinages semble être largement défavorisée par rapport à notre méthode utilisant un mouvement complexe combinant des mouvements élémentaires possiblement efficaces.

**Tableau 4.5** Comparaison des résultats du scénario 2

| Demande (connexions) | Algorithmes | Moyenne (connexions) | Ecart type (connexions) | Pourcentage |
|---|---|---|---|---|
| 100 | LONCA | 63,27 | 5,70 | 63,27% |
| | VNSFOR2 | 82,89 | 4,33 | 82,89% |
| 150 | LONCA | 84,32 | 5,65 | 56,21% |
| | VNSFOR2 | 110,25 | 5,18 | 73,50% |
| 200 | LONCA | 103,38 | 6,51 | 51,69% |
| | VNSFOR2 | 131,16 | 7,83 | 65,58% |
| 250 | LONCA | 120,34 | 7,55 | 48,13% |
| | VNSFOR2 | 149,85 | 7,99 | 59,94% |
| 300 | LONCA | 138,35 | 7,25 | 46,12% |
| | VNSFOR2 | 167,30 | 8,25 | 55,77% |
| 350 | LONCA | 153,71 | 7,68 | 43,92% |
| | VNSFOR2 | 183,42 | 9,27 | 52,41% |
| 400 | LONCA | 168,82 | 8,06 | 42,21% |
| | VNSFOR2 | 198,37 | 8,95 | 49,59% |
| 450 | LONCA | 183,69 | 8,16 | 40,82% |
| | VNSFOR2 | 216,25 | 9,27 | 48,06% |
| 500 | LONCA | 197,61 | 8,15 | 39,52% |
| | VNSFOR2 | 227,90 | 12,53 | 45,58% |

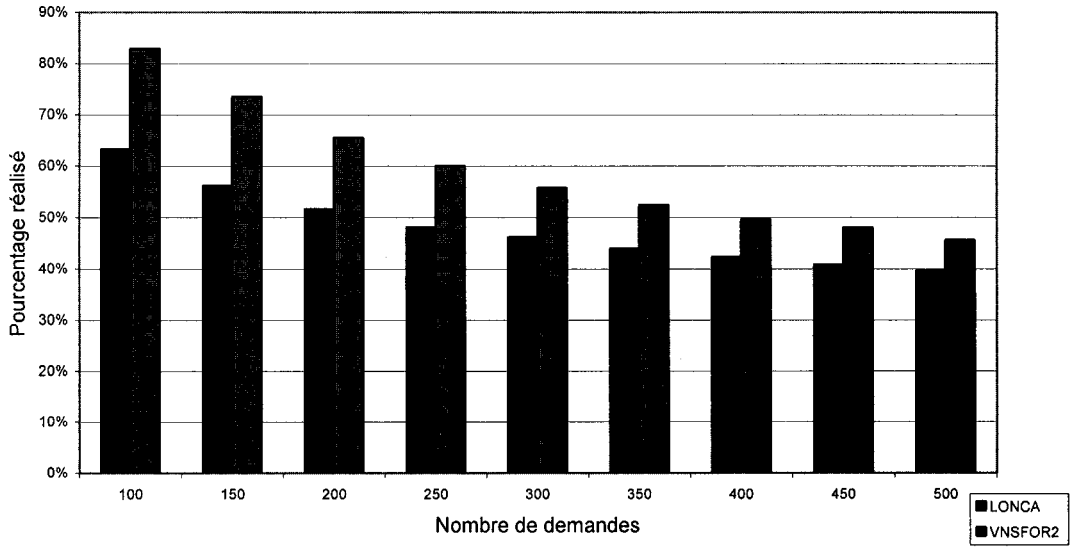



**Figure 4.9** Pourcentage de connexions réalisées, scénario 2

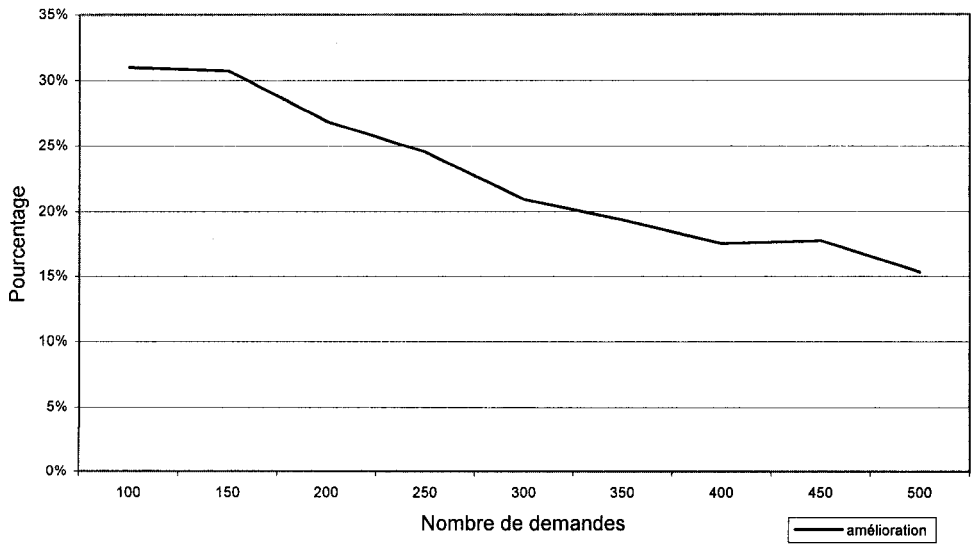



**Figure 4.10** Amélioration de VNSFOR2, scénario 2

Nous traçons à la Figure 4.10 les améliorations de l'algorithme VNSFOR2 par rapport à son homologue LONCA, selon le scénario 2. Nous notons bien l'amélioration significative pour le scénario 2 (entre 15% et 33% de connexions en plus). Cependant, la baisse de l'amélioration quand le nombre de demandes augmente peut nous mener à réfléchir. Nous constatons que la performance relative (par rapport à l'algorithme LONCA) de notre algorithme va en diminuant. Pour les cas de demandes nombreuses (au-delà de 350), nous pourrions envisager de mettre en œuvre un quatrième voisinage définissant un mouvement encore plus complexe pour résoudre ce problème plus délicat et peut-être sortir de possibles minima locaux. Il est cependant peu évident de préciser quelle est la part de connexions non réalisées car impossibles face à celles qui ne sont pas réalisées par imperfection de l'algorithme. Néanmoins, c'est face à ce scénario complexe que la performance de notre algorithme s'illustre le mieux (relativement à celle de l'algorithme LONCA une nouvelle fois).

### 4.3 Réseaux plus réalistes

Face aux résultats obtenus, nous pouvons nous interroger sur la part que possèdent les topologies des réseaux générés par rapport à la qualité des solutions exhibées. Dans un premier temps, il peut nous sembler regrettable qu'il n'existe pas de réseaux tests, parfaitement définis, pour comparer de manière exacte l'efficacité des différents algorithmes. D'autre part, nous pouvons nous questionner sur le caractère réaliste des réseaux générés selon nos critères. En se focalisant sur les réseaux générés aléatoirement, nous remarquons qu'ils possèdent de nombreuses caractéristiques absurdes :

- la connexité n'est pas assurée ;
- la 2-connexité au sens des arcs n'est pas non plus satisfaite (contrairement aux réseaux réels qui l'imposent souvent pour assurer un chemin de secours en cas de bris d'un lien du réseau) ;
- le réseau n'est pas plan (des nœuds éloignés seront rarement voisins par un lien dans la réalité, alors que c'est aléatoire dans les réseaux générés).

Symboliquement, nous pouvons voir la différence entre un réseau plan réel (le réseau NSFNET à gauche à la Figure 4.11) et une représentation d'un réseau généré aléatoirement sans contraintes d'espaces (à droite à la Figure 4.16).

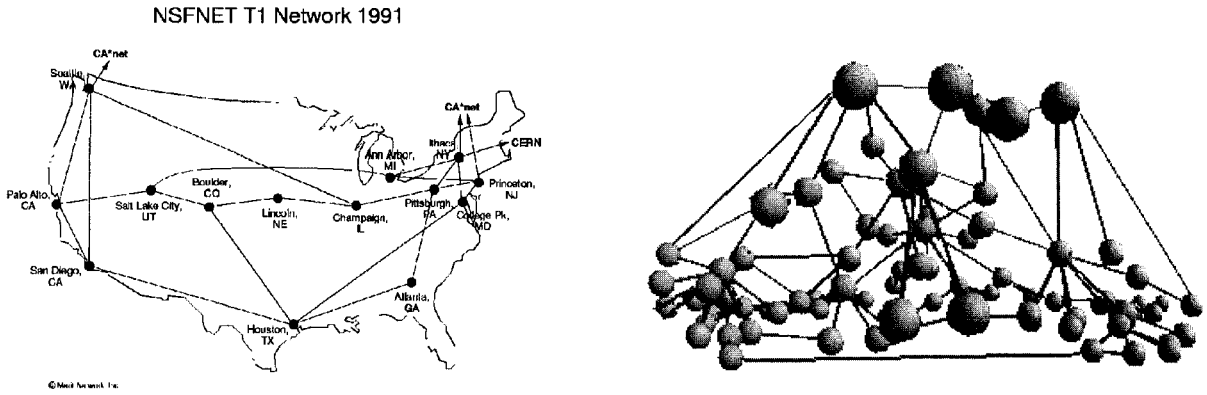



**Figure 4.11** Réseau réel plan et réseau généré

Nous avons décidé d'assurer directement la 2-connexité au sens des arcs en générant des réseaux à partir d'un squelette commun en anneau, qui comprend tous les nœuds du réseau. Nous détaillons la création d'un réseau à la Figure 4.12.

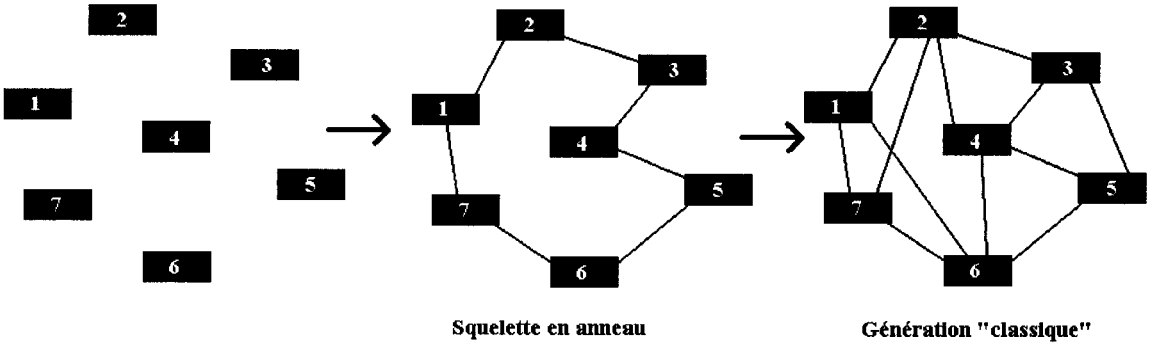



**Figure 4.12** Nouvelle méthode de génération de réseaux

À partir du squelette en anneau, nous complétons les liens manquants de la même manière que nous le faisions précédemment avec les anciens réseaux. Cette nouvelle contrainte introduite a le mérite de rendre les réseaux plus cohérents. Nous réitérons ainsi les tests précédents sur des réseaux ayant une structure sous-jacente en anneau, et nous proposons les résultats compilés dans le Tableau 4.6.

**Tableau 4.6** Comparaison des différents réseaux générés

| VNSFOR2 | Réseaux | Moyenne (connexions) | Ecart type (connexions) | Min (connexions) | Max (connexions) | Médiane (connexions) |
|---|---|---|---|---|---|---|
| 50 | "classiques" | 595,43 | 20,57 | 553 | 644 | 596 |
| | "réalistes" | 588,76 | 16,78 | 550 | 624 | 591 |
| 60 | "classiques" | 723,16 | 21,69 | 663 | 774 | 721 |
| | "réalistes" | 711,31 | 19,43 | 667 | 756 | 712 |
| 70 | "classiques" | 848,70 | 21,77 | 799 | 902 | 849 |
| | "réalistes" | 828,96 | 26,83 | 784 | 879 | 830 |
| 80 | "classiques" | 965,83 | 21,79 | 892 | 1002 | 968 |
| | "réalistes" | 951,29 | 25,82 | 875 | 988 | 948 |
| 90 | "classiques" | 1078,65 | 22,60 | 1035 | 1117 | 1082 |
| | "réalistes" | 1058,70 | 23,39 | 1016 | 1105 | 1060 |
| 100 | "classiques" | 1189,97 | 21,26 | 1142 | 1232 | 1194 |
| | "réalistes" | 1179,05 | 37,89 | 1094 | 1246 | 1176 |

Nous observons que les nouveaux résultats sont en moyenne légèrement inférieurs aux précédents. La remarque essentielle est surtout l'augmentation sensible de l'écart type dans les nouveaux réseaux avec l'augmentation de la taille de ceux-ci. On peut argumenter en premier lieu qu'il y a moins de chances dans les nouveaux réseaux que deux voisins très éloignés au niveau de l'anneau soit reliés par un même lien. Ainsi, le nombre de voisins à petite distance (via un seul routeur optique par exemple) est beaucoup plus faible dans les nouveaux réseaux, ce qui contraint d'autant le routage à optimiser. L'algorithme VNSFOR2 est, à notre avis, confronté à un environnement très contraint possédant des optima locaux d'une grande profondeur nécessitant pour s'en

extraire de mouvements d'une grande complexité. Ce type de réseaux constituerait alors une base de comparaison entre différents algorithmes très exigeante.

Une perspective ambitieuse serait de traduire le caractère plan des réseaux réels par une contrainte appropriée lors de l'affectation des voisins dans notre algorithme. Cela consisterait à identifier les contraintes spécifiques qui relient les distances entre les nœuds et la probabilité que ces nœuds soient voisins, et à les rajouter dans notre modèle initial.

# CHAPITRE V

# REFLEXIONS

Nous avons présenté jusqu'à présent les résultats finaux de notre recherche. Cependant, ils ont découlé de nombreux tests qu'ils nous a semblé judicieux de répertorier pour traduire fidèlement notre démarche tout au long de ce travail. Ces réflexions s'articulent autour de trois points centraux : détermination progressive des voisinages, paramètres choisis pour la recherche locale et tentative de résolution au moyen de la programmation par contraintes.

## 5.1 Détermination progressive des voisinages

La structure de voisinages de l'algorithme LONCA a découlé d'une série de tests à partir d'une recherche à un seul voisinage (interversion des ports de sortie). Chronologiquement, ils sont exposés ici.

### Résultats préliminaires

Tous les tests préliminaires ont été effectués sur des réseaux de 50 routeurs 8x8 (donc avec $N$=50 et $M$= 8). Nous avons défini trois algorithmes de plus en plus évolués pour réaliser ces tests préliminaires :

- le premier (cas 1) comporte le seul voisinage correspondant à l'interversion des ports de sortie ($k$=1). Le critère d'arrêt est fixé égal à 120 itérations.
- le deuxième (cas 2) est similaire au premier, mais se donne plus de temps pour trouver une bonne solution. Le critère d'arrêt est ici fixé à 200 itérations.
- le troisième (cas 3) comporte les deux voisinages $k$=1 et $k$=2 (permutation de ports en sortie, puis en entrée). Le second est réalisé si le premier n'a pas

amélioré la solution courante. Si l'un de ces voisinages n'a pas trouvé de meilleure solution, on effectue tout de même un mouvement qui laisse inchangée la fonction objectif pour ne pas boucler en cas d'échecs des deux voisinages.

Les premiers résultats comparatifs sont présentés au Tableau 5.1. Ils ont été obtenus à partir de 100 topologies différentes composées de 50 routeurs 8*8. Les premier et second cas correspondent à l'algorithme de base avec 120 et 200 itérations respectivement. Le troisième cas correspond à l'algorithme de base auquel on rajoute la possibilité de faire aussi des échanges en entrée. Le nombre d'itérations est alors fixé à 200.

**Tableau 5.1** Résultats préliminaires

| | Moyenne (connexions) | Ecart type (connexions) | Médiane (connexions) | min (connexions) | max (connexions) | Nombre de réseaux testés |
|---|---|---|---|---|---|---|
| cas 1 | 543,53 | 22,11 | 546 | 483 | 598 | 100 |
| cas 2 | 564,10 | 23,43 | 568 | 499 | 618 | 100 |
| cas 3 | 578,81 | 23,95 | 583 | 515 | 633 | 100 |
| | | | | | | |
| LONCA | 564,76 | 17,17 | 565 | 511 | 616 | 500 |

Les différentes moyennes obtenues pour tous ces algorithmes sont reportées à la Figure 5.1. Nous remarquons que nous obtenons des résultats comparables (cas 2) voire meilleurs (cas 3) que l'algorithme LONCA. Plusieurs conclusions intermédiaires s'offrent à nous. Nous notons tout d'abord que nos algorithmes ont besoin d'un nombre assez conséquent d'itérations avant de fournir de bons résultats. En effet, dans les cas 1, l'algorithme s'arrête alors qu'il est bien loin de se rapprocher du maximum atteint par le cas 2. Cela vient du fait que la recherche se fait à l'aveugle sur des petits mouvements et l'algorithme a besoin de temps pour avoir de la chance et tomber sur des mouvements efficaces.

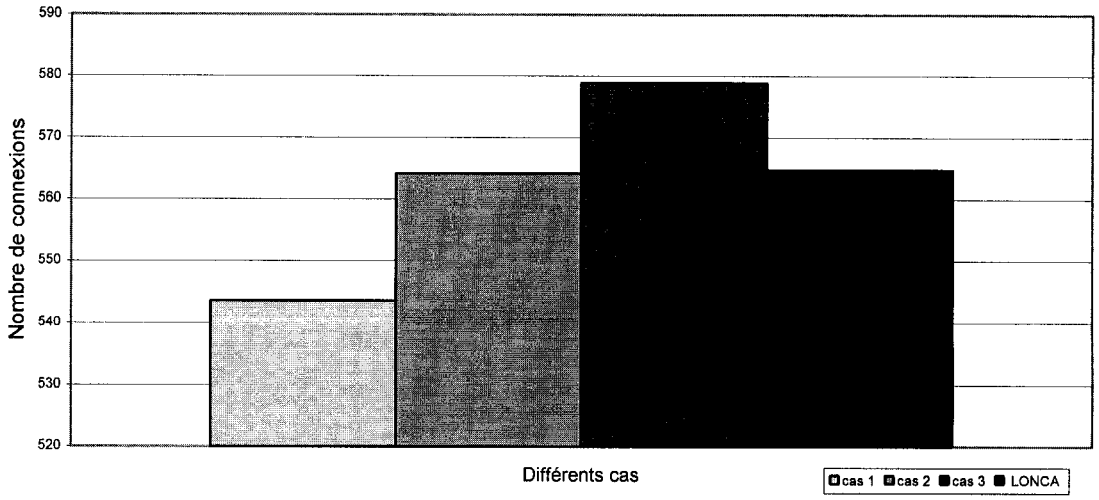



**Figure 5.1** Graphique de comparaison des algorithmes

De plus, on note que l'écart type de nos algorithmes est très important. Il est notamment préoccupant de voir que certains réseaux n'obtiennent que 75% du nombre de connexions atteints sur d'autres réseaux. Nous avons pensé initialement que les réseaux non connexes étaient la cause de ces mauvais résultats. Il n'en est quasiment rien. En testant la connexité sur les réseaux du cas 1, nous avons seulement amélioré la moyenne de 1%. Une telle approche, « à l'aveugle » et sur des petits mouvements, n'est donc absolument pas suffisante pour résoudre un problème de routage optique. Nous présentons à la Figure 5.2 l'évolution en fonction du nombre d'itérations de la qualité de la solution lors de l'exécution du dernier algorithme (cas 3).

Le temps d'exécution de cet algorithme est de deux heures environ, les 60 premières itérations étant réalisées dans les 5 premières minutes. Ainsi, pendant plus de 95% du temps, l'algorithme erre sans véritable but en espérant être chanceux dans ses choix.

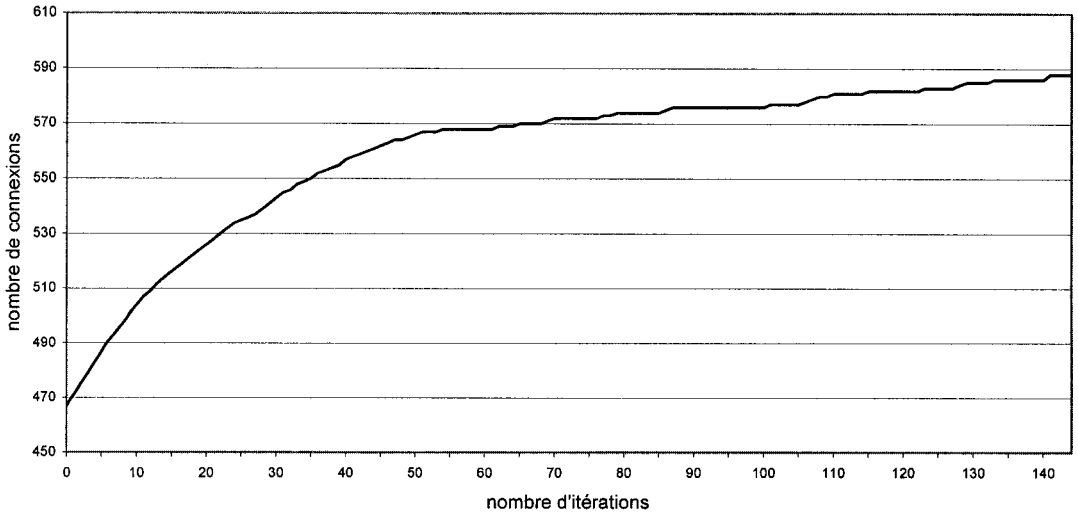



**Figure 5.2** Évolution de la qualité de la solution, cas 3 avec 50 nœuds

Notre première réflexion a été basée sur le choix des routeurs. Jusqu'à présent, ce choix était aléatoire dans notre arbre de recherche. Nous avons voulu particulariser ce choix en ordonnant les routeurs. Il fallait pour ceci introduire une fonction représentative de la qualité de branchement d'un routeur. En se basant sur le nombre théorique maximal de connexions d'un routeur (fonction de son nombre de voisins et de $M$) et du nombre effectif de connexions établies depuis ce routeur, nous avons mis en œuvre un classement des $N$ routeurs selon le rapport $\frac{cx(i)}{cx_{\max}(i)}$ croissant. ($cx(i)$ désigne ici le nombre effectif de connexions du routeur $i$ pour la solution courante, $cx_{\max}(i)$ désignant le nombre maximal de ces connexions pour ce même routeur).

Pour un routeur $i$ donné,

$$cx(i) = \sum_{j=1, j\neq i}^{N} \left(connexion[i][j]\right).$$

Un routeur $i$ qui possède $k_i$ voisins a $(M-k_i)$ ports d'entrée libres dédiés à l'émission optique, donc $(M-k_i)$ fréquences disponibles par port de sortie connecté. Ayant $k_i$ voisins, il possède $k_i$ ports de sortie connectés donc, pour un routeur $i$ donné :

$$cx_{\max}(i)=(M-k_i)k_i$$

Nous montrons un exemple de cette formule à la Figure 5.3 pour les valeurs $M$=5 et $k$=3. Nous obtenons bien $cx_{\max}(i) = (5-3).3 = 6$.

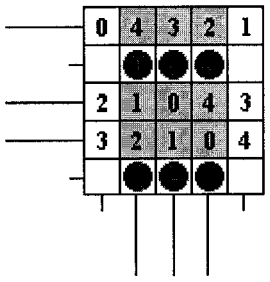



**Figure 5.3** Exemple de calcul de $cx_{\max}(i)$

Au final, les premiers routeurs de la liste créée ont des petits facteurs $\frac{cx(i)}{cx_{\max}(i)}$. Ils sont donc très loin de leur optimum personnel. Nous pensions qu'il serait intéressant de les traiter en premier. La nouvelle implémentation, incluant ce critère de sélection pour le choix du routeur à analyser, s'est trouvée être très intéressante au début de l'algorithme. Cependant, assez rapidement, à une itération donnée, on parcourt la liste des routeurs dans le même ordre que l'itération précédente. Ayant déjà été un échec l'itération antérieure, l'opération a de grandes chances d'être à nouveau inutile jusqu'à enfin essayer de nouveaux routeurs en fin de liste. Cette version n'a donc pas été retenue dan l'algorithme définitif.

Pour obtenir un algorithme très performant, il faut faire plus que seulement accélérer une recherche naïve. Il nous faut être capable d'explorer de grands voisinages. C'est alors que nous avons tenté de développer une méthode rapide pour évaluer rapidement la qualité des solutions courantes.

**Tests du re-routage**

La méthode du re-routage, ou routage incrémentiel, a été présentée dans le chapitre 3. La partie la plus ardue a été la gestion des boucles. Il fallait garder une vigilance totale sur l'état des différentes variables au moment de l'interversion des ports pour ne pas tomber dans le piège des boucles et chercher des chemins sans fin. Nous avons comparé deux algorithmes de routage à la Figure 5.4.

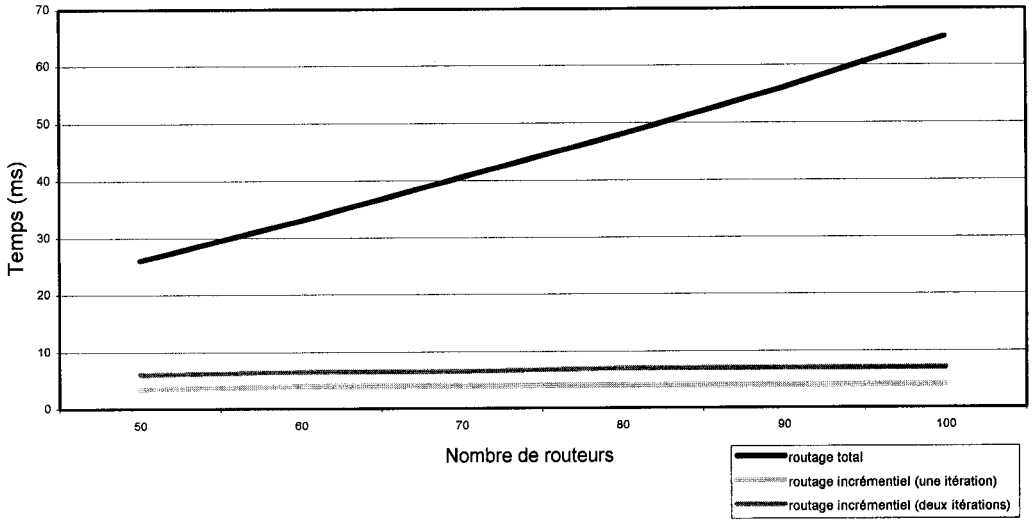



**Figure 5.4** Comparaison des algorithmes de routage

Les tests mesurés se rapportent à une permutation de ports suivi d'un routage total d'une part (courbe noire), et une permutation de ports suivi d'un re-routage d'autre part (courbes gris clair et gris foncé suivant que l'interversion mette en jeu deux ports connectés ou seul). La première remarque évidente est que la nouvelle version du routage est nettement plus rapide. Elle met au moins cinq fois moins de temps pour terminer l'exécution qui, rappelons-le, a le même effet sur les variables que le routage total. De plus, un autre point très intéressant est la non dépendance en $N$ (le nombre de routeurs) du routage incrémentiel, alors que l'ancienne version a visiblement une dépendance linéaire. De ce fait, nous serons beaucoup moins pénalisés lors du passage

des tests à des réseaux plus conséquents. Pour des réseaux de 100 routeurs, la nouvelle version est jusqu'à 10 fois plus rapide. Le troisième aspect positif de ces résultats est la distinction des deux cas : re-routage après permutation de deux ports connectés, ou après permutation d'un port connecté et d'un port non connecté. En effet, le routage incrémentiel économise des calculs inutiles lorsque la permutation ne met réellement en jeu qu'un port. Le temps est alors divisé par deux dans ce cas-ci.

Nous avons donc généré un outil performant permettant d'évaluer la qualité d'une solution de manière rapide et efficace. Ceci a rendu possible l'implémentation du troisième voisinage de la recherche locale à voisinage variable, correspondant à la permutation d'une connexion en entrée en parallèle avec la permutation d'une connexion en sortie.

## 5.2 Paramètres choisis

De nombreux paramètres de la recherche locale ont été choisis de façon empirique au cours des tests successifs et il paraît intéressant de revenir sur ces choix.

### Troisième voisinage

Nous avons tout d'abord fixé le nombre de routeurs visités à $\lfloor\sqrt{N}\rfloor$. Ce choix découlait d'un calcul sur la complexité des différents voisinages. Il pourrait être très intéressant de prolonger notre étude en mesurant l'impact concret de ce choix sur la qualité des résultats.

### Nombre d'itérations

Le choix d'un nombre d'itération variable a découlé de nos premières séries de tests comme exposé dans la partie résultats. Il peut cependant apparaître injuste le fait de se réserver la possibilité de faire varier ce nombre alors que LONCA fonctionne à nombre d'itérations constant. En fait, les auteurs de LONCA [1] stipulent que ce nombre d'itération correspond en fait au seuil au-delà duquel leur algorithme n'améliore

qu'exceptionnellement la solution courante. Ayant remarqué que notre nombre choisi initialement était trop restrictif pour certaines tailles de réseau, nous nous sommes permis de moduler le nombre d'itérations. Cette liberté nécessiterait en toute rigueur une implémentation de l'algorithme LONCA pour exhiber concrètement l'influence de l'augmentation de ce nombre d'itérations.

**Absence de « bruit »**

Une dernière remarque importante concerne une adaptation que nous avons effectué à partir de la recherche locale classique : nous avons ôté le choix aléatoire d'un voisin de la solution courante. Cependant, de nombreuses heuristiques s'attachent à introduire des facteurs d'erreurs pour éviter les optima locaux. Il semblerait très judicieux à mon avis de tester l'incorporation d'un facteur de « bruit » dans notre algorithme VNSFOR.

## 5.3 Programmation par contraintes

Une grande partie de notre recherche initiale a tenté de résoudre le présent problème à partir d'une recherche locale couplée à la programmation par contraintes, en s'inspirant des travaux de Pesant et Gendreau [19]. Nous détaillons ici les concepts que nous avons mis en place ainsi que les problèmes qui sont survenus.

La Figure 5.5 présente l'organigramme de l'algorithme qui tente de répondre à ces objectifs. Nous exposons les différentes étapes proposées, en commençant par les problèmes maître et secondaire.

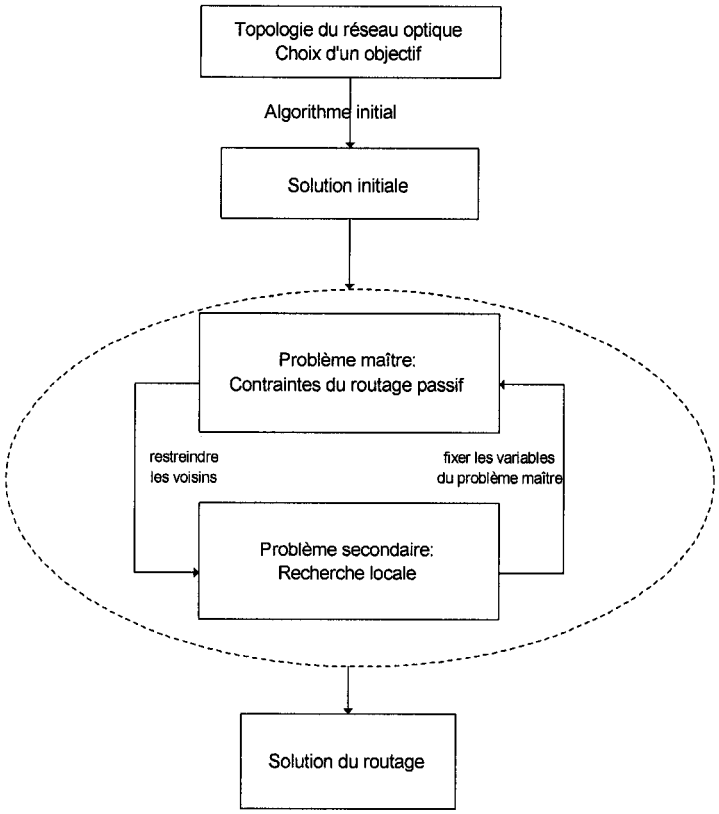



**Figure 5.5** Organigramme de l'algorithme

### 5.3.1 Présentation du problème maître

Le problème maître modélise les contraintes imposées par les routeurs passifs. D'après les branchements réalisés entre ports d'entrée et ports de sortie des différents routeurs, il évalue les connexions établies. Chaque chemin optique est seulement caractérisé par son routeur source (pour une longueur d'onde donnée, le routage passif définit alors un chemin optique unique). Le problème maître se base sur les variables suivantes :

- *Routeur_origine* (routeur $i$, port $x$, longueur d'onde $\lambda$) est égal au routeur dont est issu le chemin optique arrivant sur le port $x$ en entrée du routeur $i$ à la longueur d'onde $\lambda$;

- *Routeur_destination* (routeur $i$, port $x$, longueur d'onde $\lambda$) est égal au routeur auquel parvient le chemin optique arrivant sur le port $x$ en entrée du routeur $i$ à la longueur d'onde $\lambda$;
- *Connexion (i,j)* vaut 1 s'il existe au moins une connexion entre $i$ et $j$;
- Les connexions physiques sont effectuées avec :

$$cx\_routeur_{i,y},$$

le routeur auquel est connecté le port de sortie $y$ de $i$

$$cx\_port_{i,y},$$

le port auquel est connecté le port de sortie $y$ de $i$.

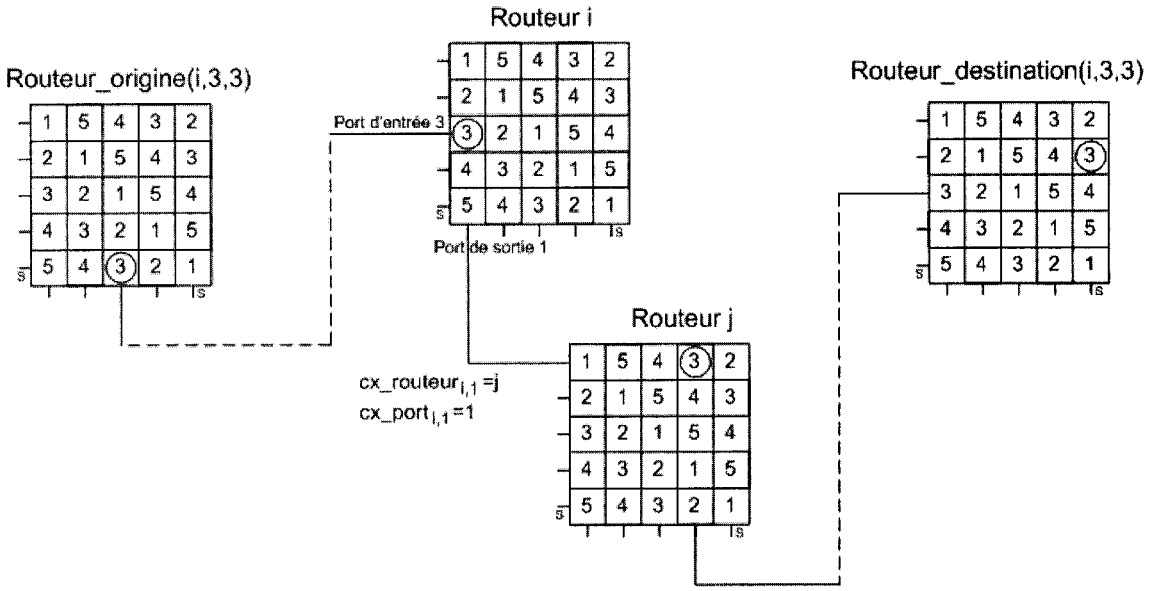



**Figure 5.6** Variables du problème maître

Toutes les variables sont présentées à la Figure 5.6. Nous y exposons aussi un exemple de branchement effectué entre les routeurs $i$ et $j$. Nous avons choisi de représenter les longueurs d'onde par des chiffres pour permettre plus tard la mise en équation des contraintes. On peut voir ici que le port de sortie 1 du routeur $i$ est relié au port d'entrée 1 du routeur $j$. Ainsi, tous les chemins optiques empruntant le port de sortie

1 de $i$ seront routés vers le port 1 du routeur $j$, et ce, quelle que soit la longueur d'onde affectée au chemin considéré.

Avec les variables ainsi proposées, nous nous devons de définir les contraintes imposées par les routeurs. Celles-ci sont au nombre de trois et correspondent au début des chemins optiques, à la terminaison de ces chemins sur les routeurs destination et aux transitions à travers les routeurs intermédiaires.

**Contrainte d'initiation de chemin optique**

Les routeurs latins imposent aux chemins optiques d'être initiés par un port d'entrée non connecté (le « port d'entrée virtuel », noté $S$, ou un port d'entrée auquel on connecte un laser émetteur pour initier un chemin optique). Cela implique donc la contrainte suivante sur la variable *Routeur_origine* du chemin considéré :

$\forall i \in [1,N], \forall \lambda \in [1,M]$ (contrainte 1)

$Routeur\_origine(i,j,\lambda) = i$ pour un port $j$ non connecté.

Nous illustrons la contrainte à la Figure 5.7. Nous notons que le port d'entrée choisi est ici le port $S$ (correspondant à la dernière ligne horizontale de la matrice de carré latin), mais il pourrait être aussi bien un autre port d'entrée non connecté. Une fois la longueur d'onde déterminée (ici 2 par exemple), le chemin optique sera alors totalement fixé grâce aux contraintes que nous allons spécifier ci-après.

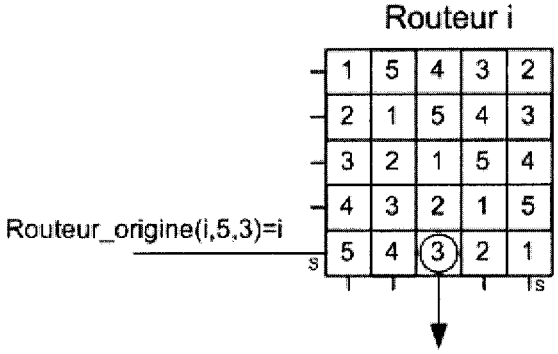



**Figure 5.7** Contrainte d'initiation

**Contrainte de terminaison de chemin optique**

Comme nous l'avons présenté dans le second chapitre, un chemin optique doit s'achever dans le réseau par une connexion qui passe par un port de sortie non connecté (le port virtuel de sortie $S$ ou un autre port non connecté auquel on connecte un récepteur de chemin optique). Cela se traduit mathématiquement par la seconde contrainte :

$\forall i \in [1,N], \forall \lambda \in [1,M]$ contrainte (2)

$Routeur\_destination(i,j,\lambda) = i$ pour un port de sortie correspondant à $j$ non connecté.

Nous reproduisons à la Figure 5.8 un exemple d'application de cette contrainte.

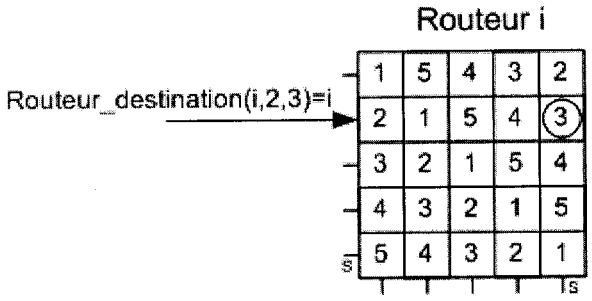



**Figure 5.8** Contrainte de terminaison

**Contrainte de transition à travers un routeur intermédiaire**

Le choix des variables que nous avons effectué nous permet de modéliser en une seule contrainte les relations imposées par le routeur latin et le routage dicté par les branchements réalisés :

$\forall i \in [1,N], \forall y \in [1,M], \forall \lambda \in [1,M]$ contrainte (3)

$$\begin{cases} routeur\_origine(cx\_routeur_{i,y}, cx\_port_{i,y}, \lambda) = routeur\_origine(i,(y+\lambda-1) \bmod M, \lambda) \\ routeur\_destination(cx\_routeur_{i,y}, cx\_port_{i,y}, \lambda) = routeur\_destination(i,(y+\lambda-1) \bmod M, \lambda) \end{cases}$$

Nous exposons un exemple d'application de cette contrainte à la Figure 5.9.

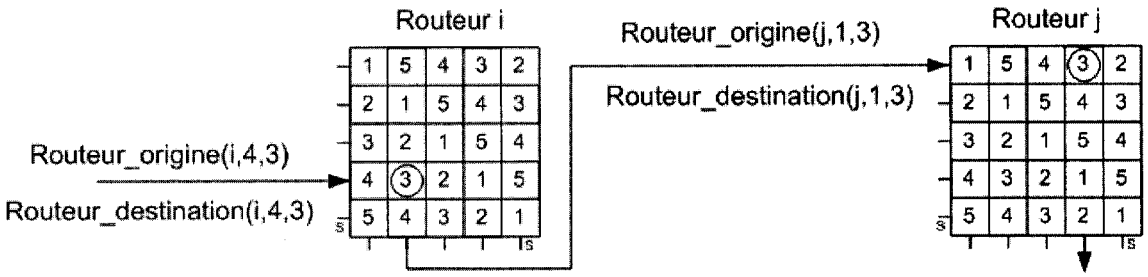



**Figure 5.9** Contrainte de transition

Nous pouvons souligner le fait, par cet exemple, que cette seule contrainte répond bien aux deux relations imposées. La présence des deux variables $cx\_routeur_{i,y}$ et $cx\_port_{i,y}$ nous assurent de faire correspondre des ports reliés entre eux par une connexion. Le membre $(y+\lambda-1) \bmod M$ traduit simplement les contraintes apportées par le routeur latin. Il découle du fait que la somme du port d'entrée sélectionné et de la longueur d'onde associée est égale au port de sortie correspondant auquel on retranche un (à un modulo $M$ près). Ainsi, le chemin optique arrivant en $j$ sur le port 3 à la longueur d'onde 2 possède le même routeur source que celui ayant atteint le routeur $i$ par le port d'entrée 4.

Le calcul des connexions est alors obtenu avec les relations :

$\forall i \in [1,N], \forall y \in [1,M], \forall \lambda \in [1,M]$

$$Connexion(routeur\_origine(i,j,\lambda), routeur\_destination(i,j,\lambda)) = 1$$

Le problème maître étant maintenant précisément introduit, nous allons définir le problème secondaire qui se charge d'étudier un voisinage de la solution courante afin de déterminer un mouvement judicieux en vue d'optimiser l'exploitation du réseau.

### 5.3.2 Présentation du problème secondaire

Le problème secondaire s'attache à définir un voisinage pour la solution courante et de mettre en place une stratégie permettant de le sonder de manière efficace. Le mouvement local de base que permettent les méthodes naïves (comme celle présentée

dans [1]) est la permutation de deux connexions (en entrée ou en sortie) d'un routeur pris au hasard dans le réseau (Figure 5.10).

La recherche locale dans le voisinage considéré se doit donc de parcourir exhaustivement la totalité de l'arbre de recherche. Il est ici de profondeur 3 (profondeur qui correspond aux trois variables de décisions : choix du routeur *i* à étudier, choix des port *j* et *k* à inverser). On a déjà coupé les branches redondantes en prenant $j<k$. Pour un parcours naïf (on se rend à tous les nœuds de l'arbre, donc on décrit tout le voisinage), on voit que l'on teste au maximum (si tous les ports sont connectés) $(M\text{-}2)+(M\text{-}3)+\ldots+1$ configurations par routeur, soit $21*N$ configurations pour les routeurs « classiques » où $M=8$.

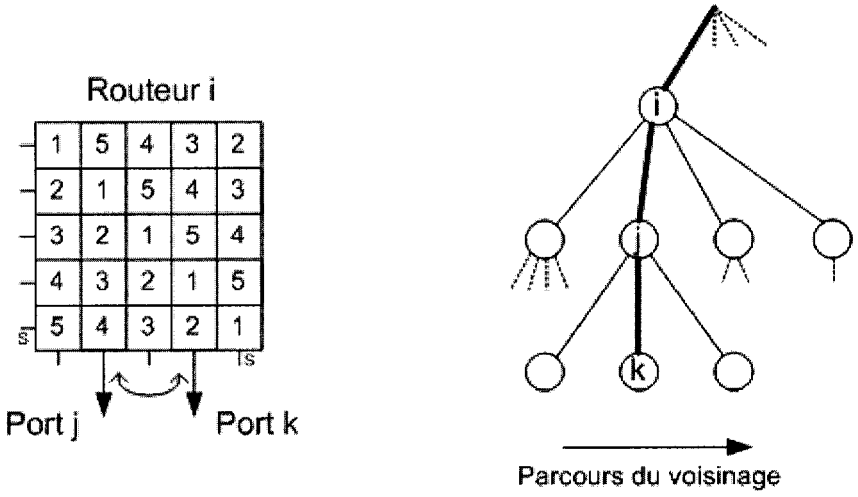



**Figure 5.10** Voisinage et arbre de recherche correspondant

Initialement, nous avons pensé que l'utilisation de la programmation par contraintes allait nous permettre de définir un voisinage beaucoup plus étendu. La recherche locale devait ainsi être beaucoup moins vulnérable aux optima locaux. Le mouvement local que nous avons alors introduit possède une profondeur de 4. Il est représenté à la Figure 5.11. Il comprend le choix d'un premier routeur (variable *i*) dans le réseau et d'un de ses ports de sortie à échanger (variable *k*), le choix d'un voisin du routeur *i* (variable *j*) et d'un de ses ports d'entrée à permuter (variable *l*). Cet arbre de

recherche est composé d'environ 220*$N$ configurations (pour $M$=8 et un nombre moyen de voisins dans le réseau pris égal à 4.5), soit plus de dix fois le voisinage précédent.

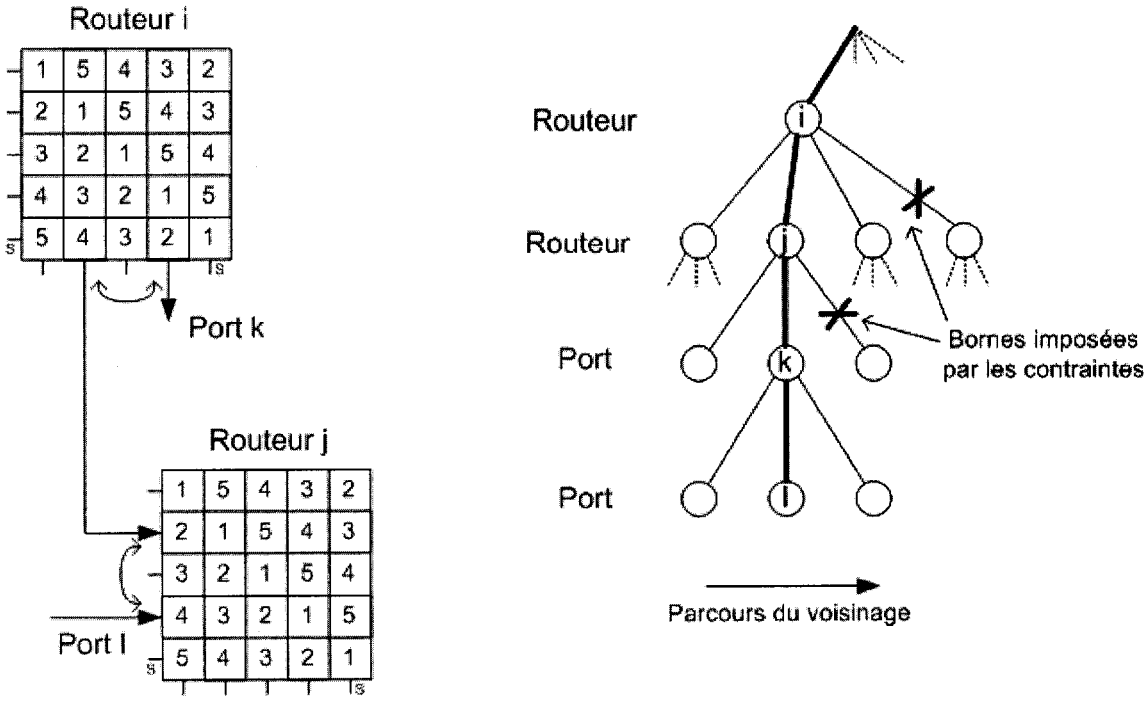



**Figure 5.11** Voisinage et arbre de recherche de notre algorithme

Il est à noter qu'une fois les routeurs choisis, le lien que nous allons faire disparaître est totalement connu étant donné qu'il n'y a qu'un seul lien entre deux routeurs connectés. Toute la difficulté du problème réside dans la création de bornes puissantes qui « défrichent » efficacement l'arbre de recherche. Nous avons été confronté ici à de sérieuses difficultés découlant de la nature même du problème. En effet, l'information obtenue sur une feuille de l'arbre ne permet pas de prédire la qualité des autres branches du même arbre (en fixant des bornes). En réalité, l'analyse d'un mouvement, au niveau du voisinage local, ne nous permet pas d'affirmer avec certitude qu'un mouvement relativement proche sera bon ou mauvais. Il y a donc très peu de propagations entre les différentes branches de l'arbre et la programmation par contraintes s'en trouve affectée. Nous avons décidé de conserver notre modèle de base

en l'exploitant avec une méthode relativement différente. En conservant cependant les mêmes notions de voisinage, nous avons choisi une approche basée sur les recherches à voisinage variable.

# CHAPITRE VI

# CONCLUSION

Le présent mémoire a porté sur le développement d'un algorithme de routage dans des réseaux comportant des routeurs latins, en absence de convertisseurs de longueurs d'onde. Nous effectuons une synthèse de nos travaux dans la première section de ce dernier chapitre. La deuxième section énonce les limitations relatives à nos méthodes ainsi qu'à leur implémentation. Enfin, nous terminons ce chapitre en énonçant dans une troisième section les travaux futurs qu'il serait possible d'entreprendre afin de poursuivre les efforts consentis pour cette recherche, dans le but d'améliorer les résultats obtenus.

## 6.1 Synthèse des travaux et contributions

Le but premier de notre recherche était de mettre au point une heuristique efficace pour optimiser le routage optique dans des réseaux de routeurs latins. Nous avons mis en œuvre un algorithme de recherche locale à voisinage variable (VNSFOR2 pour Variable Neighbourhood Search For Optical Routing 2). Les performances de celui-ci ont été comparées à un algorithme de référence de la littérature sur le sujet : l'algorithme LONCA [1]. Les résultats de notre algorithme se sont avérés meilleurs que l'algorithme de référence. De plus, dans le cas complexe du second scénario (où l'on doit répondre du mieux possible à une matrice de demande de trafic), notre algorithme obtient de bien meilleurs résultats que ceux d'une recherche naïve. L'amélioration est au moins de 4% dans le premier scénario, et au-delà de 15% dans le second scénario. Nous avons donc apporté avec notre heuristique une meilleure réponse au problème posé pour des réseaux de taille moyenne (entre cinquante et cent nœuds) et suivant les deux scénarii envisagés.

Dans un second temps, nous avons voulu tester notre algorithme sur des réseaux qui correspondraient davantage à ceux que l'on rencontre dans les cas réels. Nous avons pour ce faire créé des réseaux pseudo aléatoires : ils possèdent tous un squelette commun en anneau qui leur assure au moins une deux-connexité au sens des nœuds dans le réseau (essentiel pour assurer une meilleure survivabilité après le bris d'un lien ou d'un nœud dans le réseau) et sont ensuite générés aléatoirement. Nous avons ainsi exposé les résultats de l'algorithmes VNSFOR2 qui semblent encourageants.

## 6.2 Limitations des travaux

Nous avions comme première ambition d'exploiter la puissance de la programmation par contraintes face à des problèmes fortement contraints. Nous avons rapidement rencontré des difficultés à élaguer l'arbre de recherche des voisinages définis à partir de la propagation sur les contraintes. Nous avons donc limité l'utilisation de la programmation par contraintes à la génération de réseaux.

Nous avons aussi limité notre étude à des liens qui présente une multiplicité de un (des liens qui supportent une seule fibre optique). Nous pourrions cependant adapter notre algorithme rapidement pour qu'il puisse traiter les liens de multiplicité plus élevée. En effet, traiter des liens de multiplicité $k$ (chaque lien comprendrait $k$ fibres optiques) nécessiterait des routeurs latins de taille $7*k+1$ (dans le cas où chaque routeur a au plus 7 voisins, il faut $7*k$ ports connectables plus le port virtuel). En entrant en paramètres de l'algorithme la nouvelle taille des routeurs latins ($7*k+1$), nous serions alors capables de couvrir ce nouveau cas de figure avec l'algorithme VNSFOR2.

Finalement, nous avons introduit quelques restrictions lors de la génération de réseaux dit « réalistes ». Nous n'avons pas assuré au sens strict la survivabilité du réseau à une panne (de nœud ou de lien). Le choix d'un anneau comme squelette de base de tous les réseaux améliore la probabilité de pouvoir rerouter un chemin brisé, mais les contraintes des routeurs optiques nous empêchent de garantir que ce chemin est réalisable par une longueur d'onde. Vouloir garantir la possibilité de réparation d'une route serait un travail extrêmement fastidieux : il faudrait à la fois contrôler qu'il existe

une seconde route reliant les deux routeurs, mais aussi que cette route est disjointe d'arc (cas des pannes de liens) ou de nœuds (cas des pannes de routeurs).

### 6.3 Orientations de recherches futures

Le cas qui semble exhiber le mieux la performance de notre algorithme est le second scénario. La contrainte supplémentaire qu'impose la requête précise accroît l'écart entre les méthodes naïves et notre heuristique. Cependant, pour des réseaux de taille conséquente (à partir de 80 nœuds), nous pouvons constater un léger fléchissement de notre algorithme (en performance relative par rapport à l'algorithme LONCA). Il serait intéressant, en suivant l'idée directrice de notre mémoire, de poursuivre nos travaux en implémentant un quatrième voisinage de taille encore plus importante pour voir s'il est encore possible d'améliorer les résultats que l'on propose avec notre méthode de recherche à voisinage variable. Ce nouvel algorithme pourrait alors être testé avec notre seconde gamme de réseaux générés (dits « réalistes ») pour valider encore plus rigoureusement nos résultats.

Une autre ouverture possible de nos travaux se situe sur la génération de réseaux. Poursuivre l'amélioration du réalisme des réseaux générés est d'un intérêt indéniable pour crédibiliser les tests des différents algorithmes mis en œuvre. Ainsi, il conviendrait de s'assurer que les réseaux créés respectent du mieux possible les contraintes imposées par la disposition plane des différents nœuds. Ceci pourrait s'incorporer de manière assez naturelle à notre programme en tant que nouvelle contrainte grâce à la souplesse de la programmation par contraintes.

# BIBLIOGRAPHIE

[1] D. Banerjee et J. Frank, "Constraint Satisfaction in Optical Routing for Passive Wavelength-Routed Networks", Proc. Principles and Practice of Constraint Programming, Springer-Verlag, LNCS no. 1118, 1996, pp. 31-45.

[2] D. Banerjee et J. Frank, "Passive Optical Network Architecture based on Waveguide Grating Routers", IEEE Communications Magazine, vol. 16, no. 7, 1998, pp. 1040-1050.

[3] R. A. Barry et P. A. Humblet, « Latin Routers, Design and implementation », Journal of Lightwave Technology, vol. 11, no. 5/6, 1993, pp. 891-899.

[4] M. S. Borella et J. P. Jue, « Wavelength Routers in WDM Local Area Networks », IEEE 39th Midwest symposium on Circuits and systems, vol. 3, 1996, pp. 1202-1205.

[5] D. Brelaz, "New methods to color the vertices of a graph", Communications of the ACM, vol. 22, no. 4, 1979, pp. 251-256.

[6] M. Garnot, M. Sotom et F. Masetti, "Routing Strategies for Optical Paths in WDM Networks", IEEE Communications Magazine, vol. 1, 1997, pp. 422-426.

[7] F. Glover, "Tabu Search part I", ORSA Journal on Computing, vol. 1, no. 3, 1989, pp. 190-206

[8] F. Glover, "Tabu Search part II", ORSA Journal on Computing, vol. 2, no. 1, 1989, pp. 4-32

[9] D. Goldberg, "Genetic Algorithms", Addison Wesley editor, 1989.

[10] P. Hansen et N. Mladenovic, « Variable neighbourhood search : Principles and applications", European Journal of Operational Research, 2001, pp. 449-467.

[11] J. Holland, "Outline for a logical theory of adaptive systems", Journal of the Association of Computing Machinery, vol. 3, 1962.

[12] E. Karasan et E. Ayanoglu, "Effects of Wavelength Routing and Selection Algorithms on Wavelength Conversion Gain in WDM Optical Networks", IEEE/ACM Trans Networking, vol. 6, no. 2, Apr. 1998, pp. 186-196.

[13] S. Kirkpatrick, C. D. Gelatt Jr., et M. R. Vecchl, "Optimization by Simulated Annealing", Science, vol. 220, no. 4598, 1983, pp. 671-680.

[14] M. Kovacevic et A. Acampora, "On Wavelength Translation in All-Optical Networks", Proc. INFOCOM95, Boston, Apr. 1995, pp. 413-422.

[15] T. Lee, K. Lee, et S. Park, "Optimal routing and wavelength assignment in WDM ring networks", IEEE Journal on selected areas in communication, vol. 18, no. 10, October 2000, pp. 2146-2154.

[16] L. Li et A.K. Somani, "Dynamic wavelength routing using congestion and neighbourhood information", IEEE/ACM Trans. Networking, vol. 7, 1999, pp. 799-786.

[17] N. Mladenovic, "A variable neighbourhood algorithm: A new metaheuristic for combinatorial optimization", Abstracts of papers presented at Optimization Days, Montréal, 1995, pp. 112.

[18] A. Mokhtar et M. Azizoglu, "Adaptive wavelength routing in all-optical networks", IEEE/ACM Trans. Networking, vol. 6, no. 2, Apr 1998, pp. 197-206.

[19] G. Pesant et M. Gendreau, "A Constraint Programming Framework for Local Search Methods", Journal of Heuristics, vol. 5, 1999, pp. 255-279.

[20] R. Ramaswami et K. Sivarajan, Optical Networks: A Practical Perspective, Morgan Kaufmann Publishers, 1998.

[21] D. Saha, "Performance analysis of the effect of shorter lightpaths on the design of wavelength-routed WDM optical networks", Proc. TENCON'99, Cheju Island, vol. 1, 1999, pp. 793-796.

[22] D.A. Schupke et D. Sellier, "Lightpath Configuration of transparent and static WDM Networks for IP Traffic", IEEE Communications Magazine, vol. 2, 2001, pp. 494-498.

[23] S. Subramaniam et R. A. Barry, « Wavelength assignment in fixed-routing WDM networks", Proc. IEEE ICC, Montréal, nov. 1997, pp. 479-485.

[24] H. Waldman, D.R. Campelo et R. Camelo, "Dynamic Priority Strategies for Wavelength Asssignment in WDM Rings", Global Telecommunications Conference, GLOBECOM '00. IEEE, Vol. 2 , pp. 1288 -1292, 2000.

[25] S. Xu, L. Li et S. Wang, "Wavelength Assignment for Dynamic Traffic in WDM Networks", IEEE Communications Magazine, 2000, pp. 375-379.

[26] H. Zang, J. P. Jue, L. Sahasrabuddhe, R. Ramamurthy et B. Mukherjee, IEEE Communications Magazine, pp. 100-108, September 2001.

[27] X. Zhang et C. Qiao, "Wavelenght Assignment for Dynamic Traffic in Multi-Fiber WDM Networks", Proc. APCC/OECC'99, Beijing, Oct. 18-22, 1999, pp. 7-10.

[28] B. Zhou et H. T. Mouftah, "Adaptive Least Loaded Routing for Multi-fiber WDM Networks Using Approximate Congestion Information", IEEE Communications Magazine, vol. 5, 2002, pp. 2745-2749.

# ANNEXE

**Présentation mathématique de l'affectation de longueurs d'onde dans le cas dynamique**

Toujours dans le cadre des réseaux optiques n'ayant pas recours à la conversion de longueurs d'onde, nous présentons ici les approches conventionnelles de l'analyse dynamique du problème. Les axes de recherche évoqués privilégient également le problème d'affectation de longueurs d'onde. Ils reposent généralement sur une stratégie de routage classique et efficace : le routage fixe (FR pour Fixed Routing) basé sur le plus court chemin. Le routage est fixé une fois pour toute après la conception du réseau : à chaque paire de nœuds source-destination est affecté un chemin correspondant à la distance minimale reliant les deux nœuds.

Il convient alors de faire une description mathématique plus précise du problème pour apporter une réponse au problème dynamique d'affectation de longueurs d'onde. Considérant le réseau dans un certain état (un ensemble de connexions est déjà établi), le problème consiste à affecter une longueur d'onde à un nouvel appel de manière à minimiser la probabilité de blocage. La résolution exacte étant extrêmement difficile même en connaissant précisément le trafic, des solutions heuristiques deviennent nécessaires. Pour cela, nous définissons les variables suivantes.

- $p$ définit la requête d'appel et le chemin correspondant;
- $L(p)$ contient l'ensemble des liens du chemin $p$;
- $L_c(l,\lambda)$ est égal au nombre de fibres sur lesquelles la longueur d'onde $\lambda$ est accessible sur le lien $l$ (c'est la capacité du lien);
- $P_c(p,\lambda)$, la capacité optique du chemin $p$ (WPC pour Wavelength Path Capacity), est la capacité du lien le plus congestionné le long du chemin $p$, soit $\min_{l\in L(p)} L_c(l,\lambda)$;

- un goulot d'étranglement est un lien du chemin $p$ tel que $L_c(l,\lambda) = P_c(p,\lambda)$ ;
- $A(p)$ contient l'ensemble des longueurs d'onde accessibles pour le chemin $p$.

De manière générale, les différents algorithmes d'affectation de longueurs d'onde se chargent d'optimiser le choix de $\lambda$ dans $A(p)$ afin de minimiser la probabilité de blocage. Le choix aléatoire [14] (l'algorithme RANDOM) et le choix pré-ordonné [14] (l'algorithme FF pour First Fit) ont la particularité d'être faciles à implémenter, mais ils ne prennent pas en considération l'état du réseau et ne sont donc pas optimaux. L'algorithme de charge minimale [12] (LL pour Least Loaded) choisit la longueur d'onde qui maximise $P_c(p,\lambda)$. L'algorithme de l'utilisation maximum [18] (MU pour Most Used) choisit celle qui est la plus utilisée dans tout le réseau. Les algorithmes de la somme maximale [23] (M$\Sigma$ pour Maxsum), de la perte relative de capacité [27] (RCL pour Relative Capacity Loss) et de l'influence minimum [25] (LI pour Least Influence) tentent de sélectionner la longueur d'onde dont l'influence est minimale sur le reste du réseau, la définition de l'influence étant différente dans chacun de ces trois algorithmes.

L'algorithme de l'influence relative minimale [25] (RLI pour Relative Least Influence) est, quant à lui, indépendant de la topologie du réseau et de la matrice de trafic. Nous présentons une formulation mathématique du problème. La fonction $r_{B_c}(p,\lambda)$ évalue l'influence sur le chemin $p$ de l'affectation de $\lambda$ au chemin $p^*$.

$$r_{B_c}(p,\lambda) = \frac{B_c(p,\lambda)}{\sum_{\lambda} P_c(p,\lambda)} = \frac{\sum_{l \in L(p) \cap L(p^*)} \delta(L_c(l,\lambda), P_c(p,\lambda))}{\sum_{\lambda} P_c(p,\lambda)}$$

où $B_c(p,\lambda)$ est le nombre de liens qui sont à la fois goulots d'étranglement du chemin $p$ pour la longueur d'onde $\lambda$ et membres du chemin $p^*$,

et la fonction $\delta(L_c(l,\lambda), P_c(p,\lambda))$ est définie par :

$$\delta(L_c(l,\lambda),P_c(p,\lambda))=\begin{cases}1 & si \quad L_c(l,\lambda)=P_c(p,\lambda)\\ 0 & si \quad L_c(l,\lambda)\neq P_c(p,\lambda)\end{cases}$$

L'algorithme RLI cherche alors le minimum (sur $\lambda$) de la fonction suivante :

$$R_{B_c}(p^*,\lambda)=\sum_{p\in G(p^*)} r_{B_c}(p,\lambda)$$

La fonction $R_{B_c}(p^*,\lambda)$ représente la somme des influences relatives sur chacun des chemins $p$ qui partagent au moins un lien avec le chemin $p^*$ (décrits par la fonction $G(p^*)$).

Concrètement, l'algorithme tente de minimiser le nombre de goulots d'étranglement pondéré par la capacité optique sur leur chemin. C'est cela qui le distingue de l'algorithme LI qui ne tient pas compte de l'influence relative de cette création de goulots d'étranglement, qui minimise sur $\lambda$ la fonction :

$$\sum_{p\in G(p^*)} B_c(p,\lambda)$$

L'algorithme RLI est en fait un compromis entre le LI et l'algorithme RCL qui ne tient pas compte de la possibilité de création de plusieurs goulots d'étranglement sur le même chemin $p$, ce qui se traduit mathématiquement par minimiser sur $\lambda$ la fonction :

$$R_{CL}(p^*,\lambda)=\sum_{p\in G(p^*)} \frac{\max_{l\in L(p)\cap L(p^*)} \delta(L_c(l,\lambda),P_c(p,\lambda))}{\sum_{\lambda} P_c(p,\lambda)}$$

Une simulation a été réalisée sur un réseau particulier de 25 nœuds et les résultats obtenus ont été comparés à la limite inférieure théorique de probabilité de blocage obtenus à partir d'un réseau permettant la conversion de longueur d'onde. Que ce soit avec un routage fixé ou alternatif, RLI se comporte mieux que ses rivaux MΣ et RCL. Dans le cas du routage fixé, l'écart s'amplifie avec la largeur de bande des fibres (le

nombre $W$ de longueurs d'onde par fibre). Les résultats montrent aussi que RLI gère plus efficacement que RCL l'affectation de longueurs d'onde aux longs chemins (comprenant au moins trois sauts) car il prend en compte la possibilité d'apparition de plusieurs goulots d'étranglement et minimise ainsi la probabilité de blocage sur les longs chemins.

Le travail réalisé dans [25] présente donc un algorithme très efficace dans les conditions particulières étudiées car il donne une définition plus précise de l'influence de l'affectation de longueurs d'onde.

**Simplification du problème avec des topologies particulières**

Une autre approche possible du problème est l'étude de l'affectation de longueurs d'onde dans un réseau en anneau n'utilisant pas de convertisseur de longueur d'onde. En effet, cette topologie spécifique, rencontrée généralement dans les réseaux locaux, présente de nombreux avantages : le problème de routage se simplifie grandement grâce aux limitations imposées par la configuration physique, et l'étude du problème RWA dans sa généralité est beaucoup plus aisée; de plus, les problèmes de survie en cas de panne d'un lien sont bien plus simples à traiter. Des algorithmes ont déjà été proposés, les plus efficaces étant les algorithmes premier admissible [20, Chap. 8] (ou First Fit) qui affectent si possible des longueurs d'onde déjà utilisées dans le réseau, laissant ainsi des longueurs d'onde inexploitées, ce qui diminue les probabilités de blocage. Nous pouvons citer parmi eux, l'algorithme de priorité fixée (FP pour Fixed Priority) qui affecte la première longueur d'onde acceptable dans une liste pré-ordonnée. D'autres algorithmes favorisent l'exploitation des longueurs d'onde les plus utilisées et sont sensibles à l'état du réseau.

Nous considérons un réseau en anneau composé de fibres optiques caractérisées par $W$ longueurs d'onde permises. Le réseau est alors assimilé à $W$ sous-anneaux fonctionnant chacun sur une longueur d'onde unique. Deux nouveaux algorithmes [24] ont pour but d'améliorer les résolutions déjà existantes. Leur étude se base sur la notion de trou qu'ils définissent, à l'intérieur d'un sous-anneau, comme une séquence de liens adjacents libres pour une fréquence donnée. Un lien est libre sur un sous-anneau s'il

n'est pas utilisé à la longueur d'onde correspondante. Une requête peut alors être satisfaite s'il existe un trou qui contienne le chemin désiré. Dans le cas où au moins deux trous sont imbriqués (l'un contenant l'autre), l'allocation de la longueur d'onde du trou de longueur minimale ne modifie pas la probabilité de blocage et est donc optimale. Dans le cas de longueurs d'onde accessibles dont les trous respectifs ne sont pas imbriqués les uns dans les autres (comme présenté à la Figure 1) où $H$ correspond à la prochaine requête à satisfaire, on désigne par $a_i$ le nombre de routeurs disponibles avant le premier nœud de la requête $H$ sur le sous-anneau $i$ et par $b_i$ le nombre de routeurs disponibles après le dernier nœud de cette requête.

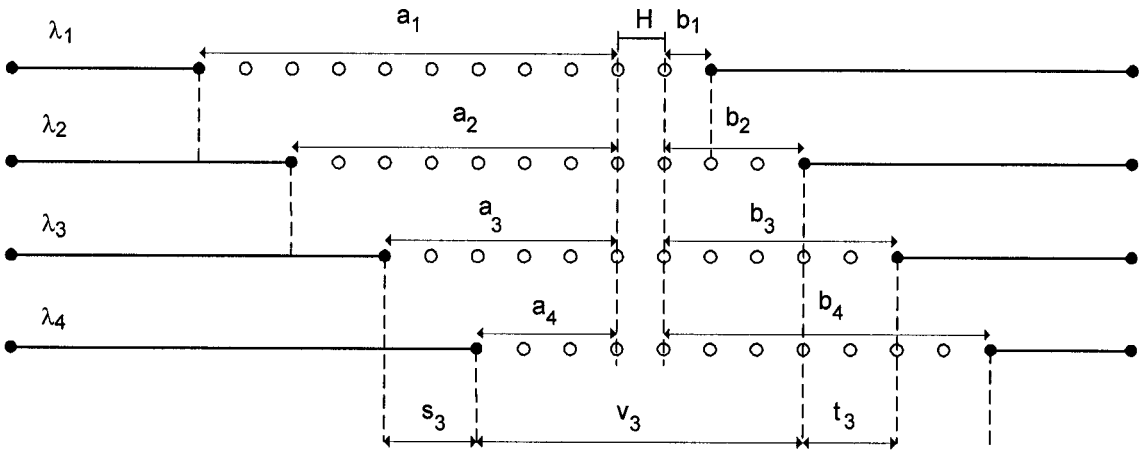



**Figure 1** Cas de trous non imbriqués

Un trait plein correspond à un chemin déjà alloué alors qu'une succession de nœuds isolés correspond à des liens libres. Nous adoptons aussi les notations suivantes :

$$s_i = a_i - a_{i+1}, \qquad i = 1,2,...,m-1$$

$$s_m = a_m + H$$

$$t_i = b_i - b_{i-1} \qquad i = 2,3,...,m$$

$t_1 = b_1 + H$

$v_i = |C_i| - s_i - t_i \qquad i = 1,2,...,m$

L'affectation qui minimise la probabilité de blocage pour la requête suivante est considérée comme étant celle qui minimise $f(s,t;v)$, définie comme la probabilité qu'un chemin optique soit requis entre un des $s$ premiers nœuds jusqu'à un des $t$ derniers, en passant à travers les $(v+1)$ nœuds centraux.

Il convient ensuite de se donner des modèles de trafic pour exploiter ces propriétés. Waldman, Campelo et Camelo [24] ont travaillé sur des trafics uniformes et exponentiels.

Pour un trafic uniforme, $f(s_i,t_i;v_i) = \frac{s_i t_i}{N^2}$ où $N$ est le nombre de nœuds de l'anneau. Minimiser $f$ revient donc à minimiser $s_i t_i$, indépendamment de $v_i$.

Dans le cas d'un trafic exponentiel où la probabilité d'avoir un chemin donné décroît exponentiellement avec la longueur de celui-ci, on a ($r$ étant une donnée comprise strictement entre 0 et 1) :

$$prob(H = i) = p(i) = (\frac{1-r}{r})r^i$$

L'espérance associée à cette probabilité est de $(\frac{1}{1-r})$. Prendre $r$ proche de 0 revient à étudier des requêtes entre nœuds très proches dans l'anneau, alors que choisir $r$ voisin de 0 traduit des requêtes entre nœuds éloignés.

Avec une telle distribution, minimiser $f$ revient à minimiser $\mu_i = r^{v_i}(1-r^{s_i})(1-r^{t_i})$. Si $r$ est proche de 1, minimiser $f$ revient encore à minimiser $s_i t_i$, alors que pour $r$ voisin de 0, minimiser $f$ s'identifie à minimiser $v_i$.

Le cas de trous imbriqués les uns dans les autres peut être fréquent si l'anneau fonctionne à des probabilités de blocage faibles (les trous sont nombreux) et les algorithmes précédents ne s'y appliquent pas de manière efficace. Il convient donc de

construire un algorithme autre que l'affectation aléatoire qui minimise non seulement la probabilité de blocage mais prépare également le réseau à la minimiser dans le futur.

Dans le cas d'un trafic uniforme, Waldman et al. [24] montrent que l'affectation qui a l'influence minimale sur la probabilité de blocage de la requête suivante est celle qui minimise $Hn_j + a_j b_j$, où $n_j$ est la longueur du trou $j$, $a_j$ et $b_j$ sont les longueurs des deux trous générés à droite et à gauche de l'ancien. Globalement, la minimisation conduit à préférer l'utilisation des petits trous, et parmi ceux-ci, ceux qui créent des configurations les plus asymétriques possibles ($a_j b_j$ faible).

Dans le cas d'un trafic exponentiel, l'optimisation consiste à minimiser $\sigma_j = r^{n_1} - r^{a_1} - r^{b_1}$ pour $r$ assez grand, et $\Delta_j = \frac{1}{N}[H + \frac{r}{1-r} + (\frac{1-r}{r})(r^{n_1} - r^{a_1} - r^{b_1})]$ sinon. Ainsi, dépendamment du facteur $r$ relatif au trafic, les facteurs importants sont la taille du trou (la plus petite possible) et l'asymétrie (qui consiste à créer un chemin le plus près possible d'un autre chemin correspondant à la même longueur d'onde).

Waldman et al. [24] proposent donc deux algorithmes qui utilisent les principes mathématiques exposés plus haut : l'algorithme de blocage minimal (MB pour Minimal Blocking) et l'algorithme de la somme maximale des capacités des routes (MC pour Maximal sum of Channel capacities). L'algorithme MB est représenté à la Figure 2.

Après avoir trié les trous par ordre de taille croissante (liste A), l'algorithme crée la liste des trous qui sont contenus dans des trous de taille supérieure (liste B). Si la liste B est vide, nous appliquons les résultats vu précédemment dans le cas de probabilités de blocage élevées (nombre de trous faibles). Si B contient un élément unique, il suffit de l'affecter. Dans le dernier cas (B contient au moins deux éléments), nous utilisons les résultats obtenus pour des probabilités de blocage faibles.

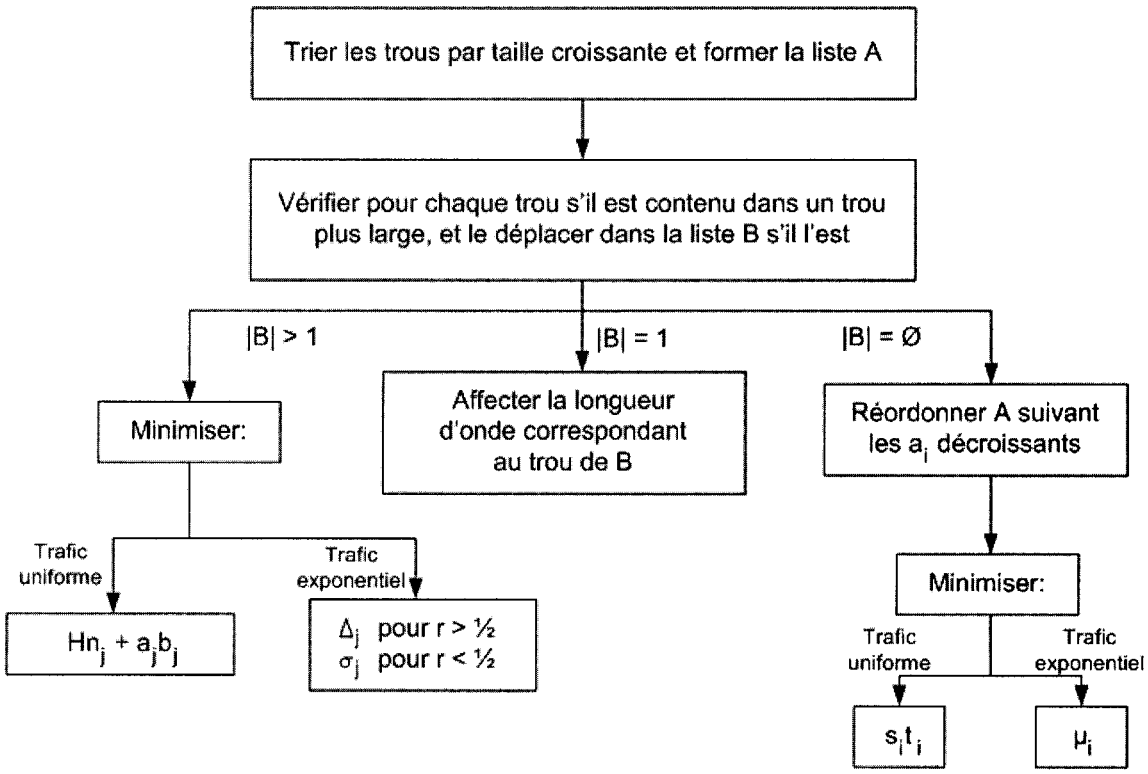



**Figure 2** Algorithme de blocage minimal (MB)

L'algorithme MC découle directement du précédent : il consiste à utiliser le traitement appliqué à B lorsque son cardinal est strictement supérieur à 1 (branche de gauche de l'arbre de la Figure 2) directement sur la liste A.

La simulation effectuée par Waldman et al. compare les performances des algorithmes RD (affectation aléatoire), FP, MC, et MB pour des trafics respectivement uniforme et exponentiel ($r$=1/2). Le réseau considéré est un anneau de 16 nœuds. Les résultats montrent une performance supérieure des algorithmes MB et MC par rapport aux autres, dans les conditions de la simulation. Les critères de petite taille des trous et d'insertion asymétrique ont donc une répercussion majeure sur la performance de l'algorithme d'affectation de longueurs d'onde.